Jims 的学习笔记

wizardforcel

Published with GitBook



目錄

Jims 的学习笔记

Awk 学习笔记

整理：Jims of 肥肥世家

jims.yang@gmail.com

第一次发布时间:2004年8月6日

Table of Contents

1. awk简介

awk是一种编程语言，用于在linux/unix下对文本和数据进行处理。数据可以来自标准输入、一个或多个文件，或其它命令的输出。它支持用户自定义函数和动态正则表达式等先进功能，是linux/unix下的一个强大编程工具。它在命令行中使用，但更多是作为脚本来使用。awk的处理文本和数据的方式是这样的，它逐行扫描文件，从第一行到最后一行，寻找匹配的特定模式的行，并在这些行上进行你想要的操作。如果没有指定处理动作，则把匹配的行显示到标准输出(屏幕)，如果没有指定模式，则所有被操作所指定的行都被处理。awk分别代表其作者姓氏的第一个字母。因为它的作者是三个人，分别是Alfred Aho、Brian Kernighan、Peter Weinberger。gawk是awk的GNU版本，它提供了Bell实验室和GNU的一些扩展。下面介绍的awk是以GUN的gawk为例的，在linux系统中已把awk链接到gawk，所以下面全部以awk进行介绍。

2. awk命令格式和选项

2.1. awk的语法有两种形式

- `awk [options] 'script' var=value file(s)`
- `awk [options] -f scriptfile var=value file(s)`

2.2. 命令选项

```
-F fs or --field-separator fs
```

指定输入文件折分隔符，fs是一个字符串或者是一个正则表达式，如-F:。

```
-v var=value or --asign var=value
```

赋值一个用户定义变量。

```
-f scripfile or --file scriptfile
```

从脚本文件中读取awk命令。

```
-mf nnn and -mr nnn
```

对nnn值设置内在限制，-mf选项限制分配给nnn的最大块数目；-mr选项限制记录的最大数目。这两个功能是Bell实验室版awk的扩展功能，在标准awk中不适用。

```
-W compact or --compat, -W traditional or --traditional
```

在兼容模式下运行awk。所以gawk的行为和标准的awk完全一样，所有的awk扩展都被忽略。

```
-W copyleft or --copyleft, -W copyright or --copyright
```

打印简短的版权信息。

```
-W help or --help, -W usage or --usage
```

打印全部awk选项和每个选项的简短说明。

```
-W lint or --lint
```

打印不能向传统unix平台移植的结构的警告。

```
-W lint-old or --lint-old
```

打印关于不能向传统unix平台移植的结构的警告。

```
-W posix
```

打开兼容模式。但有以下限制，不识别：\x、函数关键字、func、换码序列以及当fs是一个空格时，将新行作为一个域分隔符；操作符和=不能代替^和^=；fflush无效。

```
-W re-interval or --re-inerval
```

允许间隔正则表达式的使用，参考(grep中的Posix字符类)，如括号表达式[[:alpha:]]。

```
-W source program-text or --source program-text
```

使用program-text作为源代码，可与-f命令混用。

```
-W version or --version
```

打印bug报告信息的版本。

3. 模式和操作

awk脚本是由模式和操作组成的：

`pattern {action}` 如 `$ awk '/root/' test` ，或 `$ awk '$3 < 100' test` 。

两者是可选的，如果没有模式，则action应用到全部记录，如果没有action，则输出匹配全部记录。默认情况下，每一个输入行都是一条记录，但用户可通过RS变量指定不同的分隔符进行分隔。

3.1. 模式

模式可以是以下任意一个：

- /正则表达式/：使用通配符的扩展集。
- 关系表达式：可以用下面运算符表中的关系运算符进行操作，可以是字符串或数字的比较，如$2>%1选择第二个字段比第一个字段长的行。
- 模式匹配表达式：用运算符~(匹配)和~!(不匹配)。
- 模式，模式：指定一个行的范围。该语法不能包括BEGIN和END模式。
- BEGIN：让用户指定在第一条输入记录被处理之前所发生的动作，通常可在这里设置全局变量。
- END：让用户在最后一条输入记录被读取之后发生的动作。

3.2. 操作

操作由一人或多个命令、函数、表达式组成，之间由换行符或分号隔开，并位于大括号内。主要有四部份：

- 变量或数组赋值
- 输出命令
- 内置函数
- 控制流命令

4. awk的环境变量

Table 1. awk的环境变量

变量	描述
$n	当前记录的第n个字段，字段间由FS分隔。
$0	完整的输入记录。
ARGC	命令行参数的数目。
ARGIND	命令行中当前文件的位置(从0开始算)。
ARGV	包含命令行参数的数组。
CONVFMT	数字转换格式(默认值为%.6g)
ENVIRON	环境变量关联数组。
ERRNO	最后一个系统错误的描述。
FIELDWIDTHS	字段宽度列表(用空格键分隔)。
FILENAME	当前文件名。
FNR	同NR，但相对于当前文件。
FS	字段分隔符(默认是任何空格)。
IGNORECASE	如果为真，则进行忽略大小写的匹配。
NF	当前记录中的字段数。
NR	当前记录数。
OFMT	数字的输出格式(默认值是%.6g)。
OFS	输出字段分隔符(默认值是一个空格)。
ORS	输出记录分隔符(默认值是一个换行符)。
RLENGTH	由match函数所匹配的字符串的长度。
RS	记录分隔符(默认是一个换行符)。
RSTART	由match函数所匹配的字符串的第一个位置。
SUBSEP	数组下标分隔符(默认值是\034)。

5. awk运算符

Table 2. 运算符

运算符	描述
= += -= = /= %= ^= *=	赋值
:	C条件表达式
||	逻辑或
&&	逻辑与
~ ~!	匹配正则表达式和不匹配正则表达式
< <= > >= != ==	关系运算符
空格	连接
+ -	加，减
* / &	乘，除与求余
+ - !	一元加，减和逻辑非
^ ***	求幂
++ --	增加或减少，作为前缀或后缀
$	字段引用
in	数组成员

6. 记录和域

6.1. 记录

awk把每一个以换行符结束的行称为一个记录。

记录分隔符：默认的输入和输出的分隔符都是回车，保存在内建变量ORS和RS中。

$0变量：它指的是整条记录。如 `$ awk '{print $0}' test` 将输出test文件中的所有记录。

变量NR：一个计数器，每处理完一条记录，NR的值就增加1。如$ awk '{print NR,$0}' test将输出test文件中所有记录，并在记录前显示记录号。

6.2. 域

记录中每个单词称做“域”，默认情况下以空格或tab分隔。awk可跟踪域的个数，并在内建变量NF中保存该值。如 `$ awk '{print $1,$3}'` test将打印test文件中第一和第三个以空格分开的列(域)。

6.3. 域分隔符

内建变量FS保存输入域分隔符的值，默认是空格或tab。我们可以通过-F命令行选项修改FS的值。如 `$ awk -F: '{print $1,$5}' test` 将打印以冒号为分隔符的第一，第五列的内容。

可以同时使用多个域分隔符，这时应该把分隔符写成放到方括号中，如 `$awk -F'[:\t]' '{print $1,$3}' test` ，表示以空格、冒号和tab作为分隔符。

输出域的分隔符默认是一个空格，保存在OFS中。如 `$ awk -F: '{print $1,$5}' test` ，$1和$5间的逗号就是OFS的值。

7. gawk专用正则表达式元字符

一般通用的元字符集就不讲了，可参考我的Sed和Grep学习笔记。以下几个是gawk专用的，不适合unix版本的awk。

```
\Y 匹配一个单词开头或者末尾的空字符串。
\B 匹配单词内的空字符串。
\< 匹配一个单词的开头的空字符串，锚定开始。
\> 匹配一个单词的末尾的空字符串，锚定末尾。
\w 匹配一个字母数字组成的单词。
\W 匹配一个非字母数字组成的单词。
\‘ 匹配字符串开头的一个空字符串。
\' 匹配字符串末尾的一个空字符串。
```

8. POSIX字符集

可参考我的Grep学习笔记

9. 匹配操作符(~)

用来在记录或者域内匹配正则表达式。如 `$ awk '$1 ~/^root/' test` 将显示test文件第一列中以root开头的行。

10. 比较表达式

`conditional expression1 expression2: expression3` ，例如： `$ awk '{max = {$1 > $3} $1: $3: print max}' test` 。如果第一个域大于第三个域，$1就赋值给max，否则$3就赋值给max。

`$ awk '$1 + $2 < 100' test` 。如果第一和第二个域相加大于100，则打印这些行。

`$ awk '$1 > 5 && $2 < 10' test` ,如果第一个域大于5，并且第二个域小于10，则打印这些行。

11. 范围模板

范围模板匹配从第一个模板的第一次出现到第二个模板的第一次出现之间所有行。如果有一个模板没出现，则匹配到开头或末尾。如 `$ awk '/root/,/mysql/' test` 将显示root第一次出现到mysql第一次出现之间的所有行。

12. 一个验证passwd文件有效性的例子

```
$ cat /etc/passwd | awk -F: '\
NF != 7{\
printf("line %d,does not have 7 fields:%s\n",NR,$0)}\
$1 !~ /[A-Za-z0-9]/{printf("line %d,non alpha and numeric user id:%d: %s\n,NR,$0)}\
$2 == "*" {printf("line %d, no password: %s\n",NR,$0)}'
```

☐	cat把结果输出给awk，awk把域之间的分隔符设为冒号。
☐	如果域的数量(NF)不等于7，就执行下面的程序。
☐	printf打印字符串"line does not have 7 fields"，并显示该条记录。
☐	如果第一个域没有包含任何字母和数字，printf打印“no alpha and numeric user id"，并显示记录数和记录。
☐	如果第二个域是一个星号，就打印字符串“no passwd”，紧跟着显示记录数和记录本身。

13. 几个实例

- `$ awk '/^(no|so)/' test` -----打印所有以模式no或so开头的行。
- `$ awk '/^[ns]/{print $1}' test` -----如果记录以n或s开头，就打印这个记录。
- `$ awk '$1 ~/[0-9][0-9]$/(print $1}' test` -----如果第一个域以两个数字结束就打印这个记录。
- `$ awk '$1 == 100 || $2 < 50' test` -----如果第一个或等于100或者第二个域小于50，则打印该行。
- `$ awk '$1 != 10' test` -----如果第一个域不等于10就打印该行。
- `$ awk '/test/{print $1 + 10}' test` -----如果记录包含正则表达式test，则第一个域加10并打印出来。
- `$ awk '{print ($1 > 5 "ok "$1: "error"$1)}' test` -----如果第一个域大于5则打印问号后面的表达式值，否则打印冒号后面的表达式值。

- `$ awk '/^root/,/^mysql/' test`----打印以正则表达式root开头的记录到以正则表达式mysql开头的记录范围内的所有记录。如果找到一个新的正则表达式root开头的记录，则继续打印直到下一个以正则表达式mysql开头的记录为止，或到文件末尾。

14. awk编程

14.1. 变量

- 在awk中，变量不需要定义就可以直接使用，变量类型可以是数字或字符串。
- 赋值格式：`Variable = expression`，如`$ awk '$1 ~/test/{count = $2 + $3; print count}' test`,上式的作用是,awk先扫描第一个域，一旦test匹配，就把第二个域的值加上第三个域的值，并把结果赋值给变量count，最后打印出来。
- awk可以在命令行中给变量赋值，然后将这个变量传输给awk脚本。如`$ awk -F: -f awkscript month=4 year=2004 test`，上式的month和year都是自定义变量，分别被赋值为4和2004。在awk脚本中，这些变量使用起来就象是在脚本中建立的一样。注意，如果参数前面出现test，那么在BEGIN语句中的变量就不能被使用。
- 域变量也可被赋值和修改，如`$ awk '{$2 = 100 + $1; print }' test`,上式表示，如果第二个域不存在，awk将计算表达式100加$1的值，并将其赋值给$2，如果第二个域存在，则用表达式的值覆盖$2原来的值。再例如：`$ awk '$1 == "root"{$1 ="test";print}' test`，如果第一个域的值是“root”，则把它赋值为“test”，注意，字符串一定要用双引号。
- 内建变量的使用。变量列表在前面已列出，现在举个例子说明一下。`$ awk -F: '{IGNORECASE=1; $1 == "MARY"{print NR,$1,$2,$NF}'test`，把IGNORECASE设为1代表忽略大小写，打印第一个域是mary的记录数、第一个域、第二个域和最后一个域。

14.2. BEGIN模块

BEGIN模块后紧跟着动作块，这个动作块在awk处理任何输入文件之前执行。所以它可以在没有任何输入的情况下进行测试。它通常用来改变内建变量的值，如OFS,RS和FS等，以及打印标题。如：`$ awk 'BEGIN{FS=":"; OFS="\t"; ORS="\n\n"}{print $1,$2,$3} test`。上式表示，在处理输入文件以前，域分隔符(FS)被设为冒号，输出文件分隔符(OFS)被设置为制表符，输出记录分隔符(ORS)被设置为两个换行符。`$ awk 'BEGIN{print "TITLE TEST"}`只打印标题。

14.3. END模块

END不匹配任何的输入文件，但是执行动作块中的所有动作，它在整个输入文件处理完成后被执行。如 `$ awk 'END{print "The number of records is" NR}' test` ，上式将打印所有被处理的记录数。

14.4. 重定向和管道

- awk可使用shell的重定向符进行重定向输出，如： `$ awk '$1 = 100 {print $1 > "output_file" }' test` 。上式表示如果第一个域的值等于100，则把它输出到output_file中。也可以用>>来重定向输出，但不清空文件，只做追加操作。

- 输出重定向需用到getline函数。getline从标准输入、管道或者当前正在处理的文件之外的其他输入文件获得输入。它负责从输入获得下一行的内容，并给NF,NR和FNR等内建变量赋值。如果得到一条记录，getline函数返回1，如果到达文件的末尾就返回0，如果出现错误，例如打开文件失败，就返回-1。如：

  `$ awk 'BEGIN{ "date" | getline d; print d}' test` 。执行linux的date命令，并通过管道输出给getline，然后再把输出赋值给自定义变量d，并打印它。

  `$ awk 'BEGIN{"date" | getline d; split(d,mon); print mon[2]}' test` 。执行shell的date命令，并通过管道输出给getline，然后getline从管道中读取并将输入赋值给d，split函数把变量d转化成数组mon，然后打印数组mon的第二个元素。

  `$ awk 'BEGIN{while( "ls" | getline) print}'` ，命令ls的输出传递给geline作为输入，循环使getline从ls的输出中读取一行，并把它打印到屏幕。这里没有输入文件，因为BEGIN块在打开输入文件前执行，所以可以忽略输入文件。

  `$ awk 'BEGIN{printf "What is your name "; getline name < "/dev/tty" } $1 ~name {pri`
  。在屏幕上打印"What is your name ",并等待用户应答。当一行输入完毕后，getline函数从终端接收该行输入，并把它储存在自定义变量name中。如果第一个域匹配变量name的值，print函数就被执行，END块打印See you和name的值。

  `$ awk 'BEGIN{while (getline < "/etc/passwd" > 0) lc++; print lc}'` 。awk将逐行读取文件/etc/passwd的内容，在到达文件末尾前，计数器lc一直增加，当到末尾时，打印lc的值。注意，如果文件不存在，getline返回-1，如果到达文件的末尾就返回0，如果读到一行，就返回1，所以命令 `while (getline < "/etc/passwd")` 在文件不存在的情况下将陷入无限循环，因为返回-1表示逻辑真。

- 可以在awk中打开一个管道，且同一时刻只能有一个管道存在。通过close()可关闭管道。如： `$ awk '{print $1, $2 | "sort" }' test END {close("sort")}` 。awd把print语句的输出通过管道作为linux命令sort的输入,END块执行关闭管道操作。

- system函数可以在awk中执行linux的命令。如： `$ awk 'BEGIN{system("clear")'` 。

- fflush函数用以刷新输出缓冲区，如果没有参数，就刷新标准输出的缓冲区，如果以空字符串为参数，如fflush(""),则刷新所有文件和管道的输出缓冲区。

14.5. 条件语句

awk中的条件语句是从C语言中借鉴过来的，可控制程序的流程。

14.5.1. if语句

格式：

```
        {if (expression){
                   statement; statement; ...
                     }
        }
```

`$ awk '{if ($1 <$2) print $2 "too high"}' test`。如果第一个域小于第二个域则打印。

`$ awk '{if ($1 < $2) {count++; print "ok"}}' test`.如果第一个域小于第二个域，则count加一，并打印ok。

14.5.2. if/else语句，用于双重判断。

格式：

```
        {if (expression){
                   statement; statement; ...
                       }
        else{
                   statement; statement; ...
                       }
        }
```

`$ awk '{if ($1 > 100) print $1 "bad" ; else print "ok"}' test`。如果$1大于100则打印$1 bad,否则打印ok。

`$ awk '{if ($1 > 100){ count++; print $1} else {count--; print $2}' test`。如果$1大于100，则count加一，并打印$1，否则count减一，并打印$1。

14.5.3. if/else else if语句，用于多重判断。

格式：

```
{if (expression){
            statement; statement; ...
           }
else if (expression){
            statement; statement; ...
           }
else if (expression){
            statement; statement; ...
           }
else {
           statement; statement; ...
     }
}
```

14.6. 循环

- awk有三种循环:while循环；for循环；special for循环。
- `$ awk '{ i = 1; while ( i <= NF ) { print NF,$i; i++}}' test` 。变量的初始值为1，若i小于可等于NF(记录中域的个数),则执行打印语句，且i增加1。直到i的值大于NF.
- `$ awk '{for (i = 1; i<NF; i++) print NF,$i}' test` 。作用同上。
- breadkcontinue语句。break用于在满足条件的情况下跳出循环；continue用于在满足条件的情况下忽略后面的语句，直接返回循环的顶端。如：

```
{for ( x=3; x&lt;=NF; x++)
            if ($x&lt;0){print "Bottomed out!"; break}}
{for ( x=3; x&lt;=NF; x++)
            if ($x==0){print "Get next item"; continue}}
```

- next语句从输入文件中读取一行，然后从头开始执行awk脚本。如：

```
{if ($1 ~/test/){next}
    else {print}
}
```

- exit语句用于结束awk程序，但不会略过END块。退出状态为0代表成功，非零值表示出错。

14.7. 数组

awk中的数组的下标可以是数字和字母，称为关联数组。

14.7.1. 下标与关联数组

- 用变量作为数组下标。

  如： `$ awk {name[x++]=$2};END{for(i=0;i<NR;i++) print i,name[i]}' test` 。数组name中的下标是一个自定义变量x，awk初始化x的值为0，在每次使用后增加1。第二个

域的值被赋给name数组的各个元素。在END模块中，for循环被用于循环整个数组，从下标为0的元素开始，打印那些存储在数组中的值。因为下标是关健字，所以它不一定从0开始，可以从任何值开始。

- special for循环用于读取关联数组中的元素。格式如下：

```
{for (item in arrayname){
          print arrayname[item]
          }
}
```

`$ awk '/^tom/{name[NR]=$1}; END{for(i in name){print name[i]}}' test` 。打印有值的数组元素。打印的顺序是随机的。

- 用字符串作为下标。如：count["test"]

- 用域值作为数组的下标。一种新的for循环方式， `for (index_value in array) statement` 。如: `$ awk '{count[$1]++} END{for(name in count) print name,count[name]}' test` 。该语句将打印$1中字符串出现的次数。它首先以第一个域作数组count的下标，第一个域变化，索引就变化。

- delete函数用于删除数组元素。如： `$ awk '{line[x++]=$1} END{for(x in line) delete(line[x])}' test` 。分配给数组line的是第一个域的值，所有记录处理完成后，special for循环将删除每一个元素。

14.8. awk的内建函数

14.8.1. 字符串函数

- sub函数匹配记录中最大、最靠左边的子字符串的正则表达式，并用替换字符串替换这些字符串。如果没有指定目标字符串就默认使用整个记录。替换只发生在第一次匹配的时候。格式如下：

```
            sub (regular expression, substitution string):
            sub (regular expression, substitution string, target string)
```

实例：

```
            $ awk '{ sub(/test/, "mytest"); print }' testfile
            $ awk '{ sub(/test/, "mytest"); $1}; print }' testfile
```

第一个例子在整个记录中匹配，替换只发生在第一次匹配发生的时候。如要在整个文件中进行匹配需要用到gsub

第二个例子在整个记录的第一个域中进行匹配，替换只发生在第一次匹配发生的时候。

- gsub函数作用如sub，但它在整个文档中进行匹配。格式如下：

  ```
  gsub (regular expression, substitution string)
  gsub (regular expression, substitution string, target string)
  ```

  实例：

  ```
  $ awk '{ gsub(/test/, "mytest"); print }' testfile
  $ awk '{ gsub(/test/, "mytest"), $1 }; print }' testfile
  ```

  第一个例子在整个文档中匹配test，匹配的都被替换成mytest。

  第二个例子在整个文档的第一个域中匹配，所有匹配的都被替换成mytest。

- index函数返回子字符串第一次被匹配的位置，偏移量从位置1开始。格式如下：

  ```
  index(string, substring)
  ```

  实例：

  ```
  $ awk '{ print index("test", "mytest") }' testfile
  ```

  实例返回test在mytest的位置，结果应该是3。

- length函数返回记录的字符数。格式如下：

  ```
  length( string )
  length
  ```

  实例：

  ```
  $ awk '{ print length( "test" ) }'
  $ awk '{ print length }' testfile
  ```

  第一个实例返回test字符串的长度。

  第二个实例返回testfile文件中第条记录的字符数。

- substr函数返回从位置1开始的子字符串，如果指定长度超过实际长度，就返回整个字符串。格式如下：

  ```
  substr( string, starting position )
  substr( string, starting position, length of string )
  ```

  实例：

```
$ awk '{ print substr( "hello world", 7,11 ) }'
```

上例截取了world子字符串。

- match函数返回在字符串中正则表达式位置的索引，如果找不到指定的正则表达式则返回0。match函数会设置内建变量RSTART为字符串中子字符串的开始位置，RLENGTH为到子字符串末尾的字符个数。substr可利于这些变量来截取字符串。函数格式如下：

```
match( string, regular expression )
```

实例：

```
$ awk '{start=match("this is a test",/[a-z]+$/); print start}'
$ awk '{start=match("this is a test",/[a-z]+$/); print start, RSTART, RLE
```

第一个实例打印以连续小写字符结尾的开始位置，这里是11。

第二个实例还打印RSTART和RLENGTH变量，这里是11(start)，11(RSTART)，4(RLENGTH)。

- toupper和tolower函数可用于字符串大小间的转换，该功能只在gawk中有效。格式如下：

```
toupper( string )
tolower( string )
```

实例：

```
$ awk '{ print toupper("test"), tolower("TEST") }'
```

- split函数可按给定的分隔符把字符串分割为一个数组。如果分隔符没提供，则按当前FS值进行分割。格式如下：

```
split( string, array, field separator )
split( string, array )
```

实例：

```
$ awk '{ split( "20:18:00", time, ":" ); print time[2] }'
```

上例把时间按冒号分割到time数组内，并显示第二个数组元素18。

14.8.2. 时间函数

- systime函数返回从1970年1月1日开始到当前时间(不计闰年)的整秒数。格式如下：

  ```
  systime()
  ```

  实例：

  ```
  $ awk '{ now = systime(); print now }'
  ```

- strftime函数使用C库中的strftime函数格式化时间。格式如下：

  ```
  systime( [format specification][,timestamp] )
  ```

  Table 3. 日期和时间格式说明符

  ```
  | 格式 | 描述 |
  | --- | --- |
  | %a | 星期几的缩写(Sun) |
  | %A | 星期几的完整写法(Sunday) |
  | %b | 月名的缩写(Oct) |
  | %B | 月名的完整写法(October) |
  | %c | 本地日期和时间 |
  | %d | 十进制日期 |
  | %D | 日期 08/20/99 |
  | %e | 日期，如果只有一位会补上一个空格 |
  | %H | 用十进制表示24小时格式的小时 |
  | %I | 用十进制表示12小时格式的小时 |
  | %j | 从1月1日起一年中的第几天 |
  | %m | 十进制表示的月份 |
  | %M | 十进制表示的分钟 |
  | %p | 12小时表示法(AM/PM) |
  | %S | 十进制表示的秒 |
  | %U | 十进制表示的一年中的第几个星期(星期天作为一个星期的开始) |
  | %w | 十进制表示的星期几(星期天是0) |
  | %W | 十进制表示的一年中的第几个星期(星期一作为一个星期的开始) |
  | %x | 重新设置本地日期(08/20/99) |
  | %X | 重新设置本地时间(12：00：00) |
  | %y | 两位数字表示的年(99) |
  | %Y | 当前月份 |
  | %Z | 时区(PDT) |
  | %% | 百分号(%) |
  ```

  实例：

  ```
  $ awk '{ now=strftime( "%D", systime() ); print now }'
  $ awk '{ now=strftime("%m/%d/%y"); print now }'
  ```

14.8.3. 内建数学函数

Table 4.

函数名称	返回值
atan2(x,y)	y,x范围内的余切
cos(x)	余弦函数
exp(x)	求幂
int(x)	取整
log(x)	自然对数
rand()	随机数
sin(x)	正弦
sqrt(x)	平方根
srand(x)	x是rand()函数的种子
int(x)	取整，过程没有舍入
rand()	产生一个大于等于0而小于1的随机数

14.8.4. 自定义函数

在awk中还可自定义函数，格式如下：

```
        function name ( parameter, parameter, parameter, ... ) {
                        statements
                        return expression                  # the return statement and exp
        }
```

15. How-to

- 如何把一行竖排的数据转换成横排？

```
awk '{printf("%s,",$1)}' filename
```

Grep学习笔记

整理：Jims of 肥肥世家

jims.yang@gmail.com

发布时间：2004年7月16日

更新时间：2005年8月24日

Table of Contents

1. grep简介

grep（global search regular expression(RE) and print out the line,全面搜索正则表达式并把行打印出来）是一种强大的文本搜索工具，它能使用正则表达式搜索文本，并把匹配的行打印出来。Unix的grep家族包括grep、egrep和fgrep。egrep和fgrep的命令只跟grep有很小不同。egrep是grep的扩展，支持更多的re元字符，fgrep就是fixed grep或fast grep，它们把所有的字母都看作单词，也就是说，正则表达式中的元字符表示回其自身的字面意义，不再特殊。linux使用GNU版本的grep。它功能更强，可以通过-G、-E、-F命令行选项来使用egrep和fgrep的功能。

grep的工作方式是这样的，它在一个或多个文件中搜索字符串模板。如果模板包括空格，则必须被引用，模板后的所有字符串被看作文件名。搜索的结果被送到屏幕，不影响原文件内容。

grep可用于shell脚本，因为grep通过返回一个状态值来说明搜索的状态，如果模板搜索成功，则返回0，如果搜索不成功，则返回1，如果搜索的文件不存在，则返回2。我们利用这些返回值就可进行一些自动化的文本处理工作。

2. grep正则表达式元字符集（基本集）

```
^
锚定行的开始 如：'^grep'匹配所有以grep开头的行。
$
锚定行的结束 如：'grep$'匹配所有以grep结尾的行。
.
匹配一个非换行符的字符 如：'gr.p'匹配gr后接一个任意字符，然后是p。
*
匹配零个或多个先前字符 如：'*grep'匹配所有一个或多个空格后紧跟grep的行。 .*一起用代表任意字符。
[]
匹配一个指定范围内的字符，如'[Gg]rep'匹配Grep和grep。
[^]
匹配一个不在指定范围内的字符，如：'[^A-FH-Z]rep'匹配不包含A-R和T-Z的一个字母开头，紧跟rep的行。
\(..\)
标记匹配字符，如'\(love\)'，love被标记为1。
\<
锚定单词的开始，如:'\&lt;grep'匹配包含以grep开头的单词的行。
\>
锚定单词的结束，如'grep\&gt;'匹配包含以grep结尾的单词的行。
x\{m\}
重复字符x，m次，如：'0\{5\}'匹配包含5个o的行。
x\{m,\}
重复字符x,至少m次，如：'o\{5,\}'匹配至少有5个o的行。
x\{m,n\}
重复字符x，至少m次，不多于n次，如：'o\{5,10\}'匹配5--10个o的行。
\w
匹配文字和数字字符，也就是[A-Za-z0-9]，如：'G\w*p'匹配以G后跟零个或多个文字或数字字符，然后是p。
\W
\w的反置形式，匹配一个或多个非单词字符，如点号句号等。
\b
单词锁定符，如: '\bgrep\b'只匹配grep。
```

3. 用于egrep和 grep -E的元字符扩展集

```
+
匹配一个或多个先前的字符。如：'[a-z]+able'，匹配一个或多个小写字母后跟able的串，如loveable,enable,di
匹配零个或多个先前的字符。如：'gr p'匹配gr后跟一个或没有字符，然后是p的行。
a|b|c
匹配a或b或c。如：grep|sed匹配grep或sed
()
分组符号，如：love(able|rs)ov+匹配loveable或lovers，匹配一个或多个ov。
x{m},x{m,},x{m,n}
作用同x\{m\},x\{m,\},x\{m,n\}
```

4. POSIX字符类

为了在不同国家的字符编码中保持一至，POSIX(The Portable Operating System Interface)增加了特殊的字符类，如[:alnum:]是A-Za-z0-9的另一个写法。要把它们放到[]号内才能成为正则表达式，如[A-Za-z0-9]或[[:alnum:]]。在linux下的grep除fgrep外，都支持POSIX的字符类。

```
[:alnum:]
文字数字字符
[:alpha:]
文字字符
[:digit:]
数字字符
[:graph:]
非空字符（非空格、控制字符）
[:lower:]
小写字符
[:cntrl:]
控制字符
[:print:]
非空字符（包括空格）
[:punct:]
标点符号
[:space:]
所有空白字符（新行，空格，制表符）
[:upper:]
大写字符
[:xdigit:]
十六进制数字（0-9，a-f，A-F）
```

5. Grep命令选项

`-`

同时显示匹配行上下的？行，如：grep -2 pattern filename同时显示匹配行的上下2行。

`-b` ， `--byte-offset`

打印匹配行前面打印该行所在的块号码。

`-c` , `--count`

只打印匹配的行数，不显示匹配的内容。

`-f File` ， `--file=File`

从文件中提取模板。空文件中包含0个模板，所以什么都不匹配。

`-h` ，`--no-filename`

当搜索多个文件时，不显示匹配文件名前缀。

`-i` ，`--ignore-case`

忽略大小写差别。

`-q` ，`--quiet`

取消显示，只返回退出状态。0则表示找到了匹配的行。

`-l` ，`--files-with-matches`

打印匹配模板的文件清单。

`-L` ，`--files-without-match`

打印不匹配模板的文件清单。

`-n` ，`--line-number`

在匹配的行前面打印行号。

`-s` ，`--silent`

不显示关于不存在或者无法读取文件的错误信息。

`-v` ，`--revert-match`

反检索，只显示不匹配的行。

`-w` ，`--word-regexp`

如果被\<和\>引用，就把表达式做为一个单词搜索。

`-V` ，`--version`

显示软件版本信息。

6. 实例

要用好grep这个工具，其实就是要写好正则表达式，所以这里不对grep的所有功能进行实例讲解，只列几个例子，讲解一个正则表达式的写法。

`$ ls | grep '^a'`

通过管道过滤ls输出的内容，只显示以a开头的行。

`$ grep 'test' d*`

显示所有以d开头的文件中包含test的行。

`$ grep 'test' aa bb cc`

显示在aa，bb，cc文件中匹配test的行。

`$ grep '[a-z]\{5\}' aa`

显示所有包含每个字符串至少有5个连续小写字符的字符串的行。

`$ grep 'w\(es\)t.*\1' aa`

如果west被匹配，则es就被存储到内存中，并标记为1，然后搜索任意个字符（.），这些字符后面紧跟着另外一个*es*（*\1*），找到就显示该行。如果用*egrep*或*grep -E*，就不用"\"号进行转义，直接写成'*w(es)t.*\1'就可以了。

7. 技巧

- 在结果集中显示彩色的字符。

```
export GREP_OPTIONS='--color=always'
export GREP_COLOR='1;32'
```

MySQL 学习笔记

整理：Jims of 肥肥世家

yjnet@21cn.com

发布时间: 2004年11月06日

最近更新: 2005年01月12日

Table of Contents

Chapter 1. MySQL数据库的数据

MySQL数据库是由数据组成的，为了能方便管理和使用这些数据，我们把这些数据进行分类，形成各种数据类型，有数据值的类型，有表中数据列的类型，有数据表的类型。理解MySQL的这些数据类型能使我们更好地使用MySQL数据库。下面对各种数据类型进行简单的介绍。

1.1. 数据值类型(data type)

对MySQL中数据值的分类，有数值型、字符型、日期型和空值等，这和一般的编程语言的分类差不多。

1.1.1. 数值

MySQL中的数值分整型和浮点型两种。MySQL支持科学记数法。整型可以是十进制，也可是十六进制数。

1.1.2. 字符串

MySQL支持以单或双引号包围的字符序列。如“MySQL tutorial”、‘Mysql Database’。

MySQL能识别字符串中的转义序列，转义序列用反斜杠()表示。下面是一个转义序列列表。

Table 1.1. 转义序列

转义序列	含义
\0	NUL(ASCII的0值)
\'	单引号
\"	双引号
\b	后退符
\n	换行符
\r	回车符
\t	制表符
\	反斜杠
\Z	Ctrl+Z

如果字符串本身包含有单双引号，则用以下三种方法中的一种来表示：

- 字符串的引号和字符串两端的引号双同，则双写该引号。如：'mysql''s test'。
- 用与字符串的引号不同的引号把字符串引起来，如："mysql's test"。

- 用反斜杠转义引号，如："mysql\' test"，'mysql\' test'。这样就不用理会字符串两端的是单引号还是双引号了。

字符串可由一个十六进制数表示，如0x61表示字符"a"。由MySQL 4.0开始，字符串值也可用ANSI SQL表示法X'val'来表示。如X'61'表示字符"a"。

从MySQL 4.1开始，可以为字符串值专门指定一个字符集。

1.1.3. 日期和时间

MySQL默认按“年-月-日”的顺序显示日期。

1.2. 列类型(column type)

MySQL数据库的表是一个二维表，由一个或多个数据列构成。每个数据列都有它的特定类型，该类型决定了MySQL如何看待该列数据，我们可以把整型数值存放到字符类型的列中，MySQL则会把它看成字符串来处理。MySQL中的列类型有三种：数值类、字符串类和日期/时间类。从大类来看列类型和数值类型一样，都是只有三种。但每种列类型都还可细分。下面对各种列类型进行详细介绍。

1.2.1. 数值类的数据列类型

数值型的列类型包括整型和浮点型两大类。

Table 1.2. 数值类数据列类型

数据列类型	存储空间	描述
TINYINT	1字节	非常小的正整数，带符号：-128~127，不带符号：0~255
SMALLINT	2字节	小整数，带符号：-32768~32767，不带符号：0~65535
MEDIUMINT	3字节	中等大小的整数，带符号：-8388608~8388607，不带符号：0~16777215
INT	4字节	标准整数，带符号：-2147483648~2147483647，不带符号：0~4294967295
BIGINT	8字节	大整数，带符号：-9223372036854775808~9233372036854775807，不带符号：0~18446744073709551615
FLOAT	4字节	单精度浮点数，最小非零值：+-1.175494351E-38，最大非零值：+-3.402823466E+38
DOUBLE	8字节	双精度浮点数，最小非零值：+-2.2250738585072014E-308，最大非零值：+-1.7976931348623157E+308
DECIMAL	M+2字节	以字符串形式表示的浮点数，它的取值范围可变，由M和D的值决定。

1.2.1.1. 整型数据列类型

MySQL有五种整型数据列类型，即TINYINT，SMALLINT，MEDIUMINT，INT和BIGINT。它们之间的区别是取值范围不同，存储空间也各不相同。在整型数据列后加上UNSIGNED属性可以禁止负数，取值从0开始。

声明整型数据列时，我们可以为它指定个显示宽度M(1~255)，如INT(5)，指定显示宽度为5个字符,如果没有给它指定显示宽度，MySQL会为它指定一个默认值。显示宽度只用于显示，并不能限制取值范围和占用空间，如：INT(3)会占用4个字节的存储空间，并且允许的最大值也不会是999,而是INT整型所允许的最大值。

1.2.1.2. 浮点型数据列类型

MySQL有三种浮点型数据列类型，分别是：FLOAT，DOUBLE和DECIMAL。浮点类数据类型有一个最大可表示值和一个最小非零可表示值，最小非零可表示值决定了该类型的精确度。

MySQL 4.0.2版之后，FLOAT和DOUBLE都可以指定UNSIGNED属性。当指定该属性时，取值范围不平移到正数区间，而只是简单地把浮点类型的负数部份去掉。

浮点类型也有M(1~255)和D(1~30，且不能大于M-2)。分别表示显示宽度和小数位数。M和D在FLOAT和DOUBLE中是可选的，默认，当MySQL版本大于3.23.6时，FLOAT和DOUBLE类型将被保存为硬件所支持的最大精度。DECIMAL的M和D值在MySQL3.23.6后可选，默认D值

为0,M值为10。

1.2.1.3. 如何选择数值类数据列类型？

为了节省存储空间和提高数据库处理效率，我们应根据应用数据的取值范围来选择一个最适合的数据列类型。如果把一个超出数据列取值范围的数存入该列，则MySQL就会截短该值，如：我们把99999存入SMALLINT(3)数据列里，因为SMALLINT(3)的取值范围是-32768~32767，所以就会被截短成32767存储。显示宽度3不会影响数值的存储。只影响显示。

对于浮点数据列，存入的数值会被该列定义的小数位进行四舍五入。如把一个1.234存入FLOAT(6.1)数据列中，结果是1.2。

DECIMAL与FLOAT和DOUBLE的区别是：DECIMAL类型的值是以字符串的形式被储存起来的，它的小数位数是固定的。它的优点是，不会象FLOAT和DOUBLE类型数据列那样进行四舍五入而产生误差，所以很适合用于财务计算；而它的缺点是：由于它的存储格式不同，CPU不能对它进行直接运算，从而影响运算效率。DECIMAL(M，D)总共要占用M+2个字节。

1.2.1.4. 数值类数据列的属性

- ZEROFILL属性适用于所有数值类数据列类型，作用是，如果数值的宽度小于定义的显示宽度，则在数值前填充0。
- UNSIGNED属性不允许数据列出现负数。
- AUTO_INCREMENT属性可生成独一无二的数字序列。只对整数类的数据列有效。
- NULL和NOT NULL属性设置数据列是否可为空。
- DEFAULT属性可为数据列指定默认值。

1.2.2. 字符串类数据列类型

字符串可以用来表示任何一种值，所以它是最基本的类型之一。我们可以用字符串类型来存储图象或声音之类的二进制数据，也可存储用gzip压缩的数据。下表介绍了各种字符串类型：

Table 1.3. 字符串类数据列类型

类型	最大长度	占用存储空间
CHAR[(M)]	M字节	M字节
VARCHAR[(M)]	M字节	L+1字节
TINYBLOD，TINYTEXT	2^8-1字节	L+1字节
BLOB，TEXT	2^16-1字节	L+2
MEDIUMBLOB，MEDIUMTEXT	2^24-1字节	L+3
LONGBLOB，LONGTEXT	2^32-1字节	L+4
ENUM('value1','value2',...)	65535个成员	1或2字节
SET('value1','value2',...)	64个成员	1,2,3,4或8字节

L+1、L+2是表示数据列是可变长度的，它占用的空间会根据数据行的增减面则改变。数据行的总长度取决于存放在这些数据列里的数据值的长度。L+1或L+2里多出来的字节是用来保存数据值的长度的。在对长度可变的数据进行处理时，MySQL要把数据内容和数据长度都保存起来。

如果把超出字符串最大长度的数据放到字符类数据列中，MySQL会自动进行截短处理。

ENUM和SET类型的数据列定义里有一个列表，列表里的元素就是该数据列的合法取值。如果试图把一个没有在列表里的值放到数据列里，它会被转换为空字符串(“”)。

字符串类型的值被保存为一组连续的字节序列，并会根据它们容纳的是二进制字符串还是非二进制字符而被区别对待为字节或者字符：

- 二进制字符串被视为一个连续的字节序列，与字符集无关。MySQL把BLOB数据列和带BINARY属性的CHAR和VARCHAR数据列里的数据当作二进制值。
- 非二进制字符串被视为一个连续排列的字符序列。与字符集有关。MySQL把TEXT列与不带BINARY属性的CHAR和VARCHAR数据列里的数据当作二进制值对待。

在MySQL4.1以后的版本中，不同的数据列可以使用不同的字符集。在MySQL4.1版本以前，MySQL用服务器的字符集作为默认字符集。

非二进制字符串，即我们通常所说的字符串，是按字符在字符集中先后次序进行比较和排序的。而二进制字符串因为与字符集无关，所以不以字符顺序排序，而是以字节的二进制值作为比较和排序的依据。下面介绍两种字符串的比较方式：

- 二进制字符串的比较方式是一个字节一个字节进行的，比较的依据是两个字节的二进制值。也就是说它是区分大小写的，因为同一个字母的大小写的数值编码是不一样的。
- 非二进制字符串的比较方式是一个字符一个字符进行的，比较的依据是两个字符在字符集中的先后顺序。在大多数字符集中，同一个字母的大小写往往有着相同的先后顺序，所以它不区分大小写。

二进制字符串与字符集无关，所以无论按字符计算还是按字节计算，二进制字符串的长度都是一样的。所以VARCHAR(20)并不表示它最多能容纳20个字符，而是表示它最多只能容纳可以用20个字节表示出来的字符。对于单字节字符集，每个字符只占用一个字节，所以这两者的长度是一样的，但对于多字节字符集，它能容纳的字符个数肯定少于20个。

1.2.2.1. CHAR和VARCHAR

CHAR和VARCHAR是最常用的两种字符串类型，它们之间的区别是：

- CHAR是固定长度的，每个值占用相同的字节，不够的位数MySQL会在它的右边用空格字符补足。
- VARCHAR是一种可变长度的类型，每个值占用其刚好的字节数再加上一个用来记录其长度的字节即L+1字节。

CHAR(0)和VARCHAR(0)都是合法的。VARCHAR(0)是从MySQL4.0.2版开始的。它们的作用是作为占位符或用来表示各种on/off开关值。

如何选择CHAR和VARCHAR，这里给出两个原则：

- 如果数据都有相同的长度，选用VARCHAR会多占用空间，因为有一位用来存储其长度。如果数据长短不一，选用VARCHAR能节省存储空间。而CHAR不论字符长短都需占用相同的空间，即使是空值也不例外。
- 如果长度出入不大，而且是使用MyISAM或ISAM类型的表，则用CHAR会比VARCHAR好，因为MyISAM和ISAM类型的表对处理固定长度的行的效率高。

> 在一个数据表里，只要有一个数据列的长度是可变的，则所有数据列的长度将是可变的。MySQL会进行自动地转换。一个例外，CHAR长度小于4的不会进行自动转换，因为MySQL会认为这样做没必要，节省不了多少空间。反而MySQL会把大量长度小的VARCHAR转换成CHAR，以减少空间占用量。

1.2.2.2. BLOB和TEXT

BLOB是二进制字符串，TEXT是非二进制字符串。两者都可存放大容量的信息。

有关BLOB和TEXT索引的建立：

- BDB表类型和MySQL3.23.2以上版本的MyISAM表类型允许在BLOB和TEXT数据列上建立索引。
- ISAM、HEAP和InnoDB表不支持大对象列的索引。

使用BLOB和TEXT应注意的问题：

- 由于这两个列类型所存储的数据量大，所以删除和修改操作容易在数据表里产生大量的碎片，需定期运行OPTIMIZE TABLE以减少碎片和提高性能。

- 如果使用的值非常巨大，就需对服务器进行相应的优化调整，增加max_allowed_packet参数的值。对那些可会用到变些巨大数据的客户程序，也需加大它们的数据包大小。

1.2.2.3. ENUM和SET

ENUM和SET都是比较特殊的字符串数据列类型，它们的取值范围是一个预先定义好的列表。ENUM或SET数据列的取值只能从这个列表中进行选择。ENUM和SET的主要区别是：

- ENUM只能取单值，它的数据列表是一个枚举集合。它的合法取值列表最多允许有65535个成员。例如：ENUM("N","Y")表示，该数据列的取值要么是"Y"，要么就是"N"。
- SET可取多值。它的合法取值列表最多允许有64个成员。空字符串也是一个合法的SET值。

ENUM和SET的值是以字符串形式出现的，但在内部，MySQL以数值的形式存储它们。

- ENUM的合法取值列表中的字符串被按声明定义的顺序被编号，从1开始。
- SET的编号不是按顺序进行编号的，SET中每一个合法取值都对应着SET值里的一个位。第一个合法取值对应0位，第二个合法取值对应1位，以此类推，如果数值形式的SET值等于0,则说明它是一个空字符串，如果某个合法的取值出现在SET数据列里，与之对应的位就会被置位；如果某个合法的取值没有出现在SET数据列里，与之对应的位就会被清零。正因为SET值与位有这样的对应关系，所以SET数据列的多个合法取值才能同时出现并构成SET值。

1.2.2.4. 字符串类型数据列的字符集属性

在MySQL 4.1以前的版本，字符串数据列的字符集由服务器的字符决定，MySQL 4.1版以后的版本可对每个字符串数据列指定不同的字符串。如果按默认方式设置，可按数据列、数据表、数据库、服务器的顺序关联字符串的字符集，直到找一个明确定义的字符集。

1.2.3. 日期,时间型数据列类型

MySQL的日期时间类型有：DATE，DATETIME，TIME，TIMESTAMP和YEAR，下表是这些类型的取值范围和存储空间要求：

Table 1.4. 日期，时间类型列

类型	取值范围	存储空间	零值表示法
DATE	1000-01-01~9999-12-31	3字节(MySQL3.23版以前是4字节)	0000-00-00
TIME	-838:59:59~838:59:59	3字节	00:00:00
DATETIME	1000-01-01 00:00:00~9999-12-31 23:59:59	8字节	0000-00-00 00:00:00
TIMESTAMP	19700101000000~2037年的某个时刻	4字节	00000000000000
YEAR	YEAR(4)：1901~2155 YEAR(2)：1970~2069	1字节	0000

> MySQL总是把日期和日期里的年份放在最前面，按年月日的顺序显示。

1.2.3.1. DATE、TIME、DATATIME数据列类型

DATE、TIME和DATATIME类型分别存放日期值、时间值、日期和时间值的组合。它们的格式分别是“CCYY-MM-DD”、“hh:mm:ss”、“CCYY-MM-DD hh:mm:ss”。

DATATIME里的时间值和TIME值是有区别的，DATATIME里的时间值代表的是几点几分，TIME值代表的是所花费的时间。当向TIME数据列插值时，需用时间的完整写法，如12分30秒要写成“00:12:30”。

1.2.3.2. TIMESTAMP数据列类型

TIMESTAMP数据列的格式是CCYYMMDDhhmmss，取值范围从19700101000000开始，即1970年1月1号，最大到2037年。它的特点是能把数据行的创建或修改时间记录下来：

- 如果把一个NULL值插入TIMESTAMP列，这个数据列就将自动取值为当前的日期和时间。
- 在创建和修改数据行时，如果没有明确对TIMESTAMP数据列进行赋值，则它就会自动取值为当前的日期和时间。如果行中有多个TIMESTAMP列，只有第一个会自动取值。
- 如果对TIMESTAMP设置一个确定的日期和时间值，则会使TIMESTAMP的自动取值功能失效。

TIMESTAMP默认的列宽是14,可指定列宽，以改变显示效果。但不论你指定的列宽如何，MySQL都是以4字节来存储TIMESTAMP值，也总是以14位精度来计算。

如果需要把创建时间和最近一次修改时间同时记录下来，可以用两个时间戳来记录，一个记录创建时间，一个记录修改时间。不过需记住两件事，一是要把记录修改时间的TIMESTAMP数据列放在最前面，这样才会自动取值；二是创建一条新记录时，要用now()函数来初始化创建时间TIMESTAMP数据列，这样，该TIMESTAMP数据列就不会再变化。

1.2.3.3. YEAR

YEAR是一种单字节的数据列类型，YEAR(4)的取值范围是1901~2155,YEAR(2)的取值范围是1970~2069,但只显示最后两位数。MySQL能自动把两位数字年份转换成四位数字的年份，如97和14分被转换成1997和2014。转换规则是这样的：

- 年份值00~69将被转换成2000~2069；
- 年份值70~99将被转换成1970~1999。

> 00被转换成0000,而不是2000。因为数值00也就是0,而0值是YEAR的一个合法取值。

1.3. 唯一编号

在数据库应用，我们经常要用到唯一编号，以标识记录。在MySQL中可通过数据列的AUTO_INCREMENT属性来自动生成。MySQL支持多种数据表，每种数据表的自增属性都有差异，这里将介绍各种数据表里的数据列自增属性。

- ISAM表
    - 如果把一个NULL插入到一个AUTO_INCREMENT数据列里去，MySQL将自动生成下一个序列编号。编号从1开始，并1为基数递增。
    - 把0插入AUTO_INCREMENT数据列的效果与插入NULL值一样。但不建议这样做，还是以插入NULL值为好。
    - 当插入记录时，没有为AUTO_INCREMENT明确指定值，则等同插入NULL值。
    - 当插入记录时，如果为AUTO_INCREMENT数据列明确指定了一个数值，则会出现两种情况，情况一，如果插入的值与已有的编号重复，则会出现出错信息，因为AUTO_INCREMENT数据列的值必须是唯一的；情况二，如果插入的值大于已编号的值，则会把该插入到数据列中，并使在下一个编号将从这个新值开始递增。也就是说，可以跳过一些编号。
    - 如果自增序列的最大值被删除了，则在插入新记录时，该值被重用。
    - 如果用UPDATE命令更新自增列，如果列值与已有的值重复，则会出错。如果大于已有值，则下一个编号从该值开始递增。
    - 如果用replace命令基于AUTO_INCREMENT数据列里的值来修改数据表里的现有记录，即AUTO_INCREMENT数据列出现在了replace命令的where子句里，相应的AUTO_INCREMENT值将不会发生变化。但如果replace命令是通过其它的PRIMARY KEY OR UNIQUE索引来修改现有记录的(即AUTO_INCREMENT数据列没有出现在replace命令的where子句中)，相应的AUTO_INCREMENT值--如果设置其为NULL(如没有对它赋值)的话--就会发生变化。

  - last_insert_id()函数可获得自增列自动生成的最后一个编号。但该函数只与服务器的本次会话过程中生成的值有关。如果在与服务器的本次会话中尚未生成AUTO_INCREMENT值，则该函数返回0。

其它数据表的自动编号机制都以ISAM表中的机制为基础。

- MyISAM数据表
  - 删除最大编号的记录后，该编号不可重用。
  - 可在建表时可用“AUTO_INCREMENT=n”选项来指定一个自增的初始值。
  - 可用alter table table_name AUTO_INCREMENT=n命令来重设自增的起始值。
  - 可使用复合索引在同一个数据表里创建多个相互独立的自增序列，具体做法是这样的：为数据表创建一个由多个数据列组成的PRIMARY KEY OR UNIQUE索引，并把AUTO_INCREMENT数据列包括在这个索引里作为它的最后一个数据列。这样，这个复合索引里，前面的那些数据列每构成一种独一无二的组合，最末尾的AUTO_INCREMENT数据列就会生成一个与该组合相对应的序列编号。
- HEAP数据表
  - HEAP数据表从MySQL4.1开始才允许使用自增列。
  - 自增值可通过CREATE TABLE语句的 AUTO_INCREMENT=n选项来设置。
  - 可通过ALTER TABLE语句的AUTO_INCREMENT=n选项来修改自增始初值。
  - 编号不可重用。
  - HEAP数据表不支持在一个数据表中使用复合索引来生成多个互不干扰的序列编号。
- BDB数据表
  - 不可通过CREATE TABLE OR ALTER TABLE的AUTO_INCREMENT=n选项来改变自增初始值。
  - 可重用编号。
  - 支持在一个数据表里使用复合索引来生成多个互不干扰的序列编号。
- InnDB数据表
  - 不可通过CREATE TABLE OR ALTER TABLE的AUTO_INCREMENT=n选项来改变自增初始值。
  - 不可重用编号。
  - 不支持在一个数据表里使用复合索引来生成多个互不干扰的序列编号。

在使用AUTO_INCREMENT时，应注意以下几点：

- AUTO_INCREMENT是数据列的一种属性，只适用于整数类型数据列。
- 设置AUTO_INCREMENT属性的数据列应该是一个正数序列，所以应该把该数据列声明为UNSIGNED，这样序列的编号个可增加一倍。
- AUTO_INCREMENT数据列必须有唯一索引，以避免序号重复。
- AUTO_INCREMENT数据列必须具备NOT NULL属性。
- AUTO_INCREMENT数据列序号的最大值受该列的数据类型约束，如TINYINT数据列的最大编号是127,如加上UNSIGNED，则最大为255。一旦达到上限，AUTO_INCREMENT就会失效。
- 当进行全表删除时，AUTO_INCREMENT会从1重新开始编号。全表删除的意思是发出以下两条语句时：

  ```
  delete from table_name;
  or
  truncate table table_name
  ```

  这是因为进行全表操作时，MySQL实际是做了这样的优化操作：先把数据表里的所有数据和索引删除，然后重建数据表。如果想删除所有的数据行又想保留序列编号信息，可这样用一个带where的delete命令以抑制MySQL的优化：

  ```
  delete from table_name where 1;
  ```

  这将迫使MySQL为每个删除的数据行都做一次条件表达式的求值操作。

- 强制MySQL不复用已经使用过的序列值的方法是：另外创建一个专门用来生成AUTO_INCREMENT序列的数据表，并做到永远不去删除该表的记录。当需要在主数据表里插入一条记录时，先在那个专门生成序号的表中插入一个NULL值以产生一个编号，然后，在往主数据表里插入数据时，利用LAST_INSERT_ID()函数取得这个编号，并把它赋值给主表的存放序列的数据列。如：

  ```
  insert into id set id = NULL;
  insert into main set main_id = LAST_INSERT_ID();
  ```

- 可用alter命令给一个数据表增加一个具有AUTO_INCREMENT属性的数据列。MySQL会自动生成所有的编号。
- 要重新排列现有的序列编号，最简单的方法是先删除该列，再重建该，MySQL会重新生连续的编号序列。

- 在不用AUTO_INCREMENT的情况下生成序列，可利用带参数的LAST_INSERT_ID()函数。如果用一个带参数的LAST_INSERT_ID(expr)去插入或修改一个数据列，紧接着又调用不带参数的LAST_INSERT_ID()函数，则第二次函数调用返回的就是expr的值。下面演示该方法的具体操作：

  先创建一个只有一个数据行的数据表：

  ```
  create table seq_table (id int unsigned not null);
  insert into seq_table values (0);
  ```

  接着用以下操作检索出序列号：

  ```
  update seq_table set seq = LAST_INSERT_ID( seq + 1 );
  select LAST_INSERT_ID();
  ```

  通过修改seq+1中的常数值，可生成不同步长的序列，如seq+10可生成步长为10的序列。

  该方法可用于计数器，在数据表中插入多行以记录不同的计数值。再配合LAST_INSERT_ID()函数的返回值生成不同内容的计数值。这种方法的优点是不用事务或LOCK，UNLOCK表就可生成唯一的序列编号。不会影响其它客户程序的正常表操作。

1.4. 字符集支持

MySQL4.1以前版本服务器只能使用单一字符集，从MySQL4.1版本开始，不仅服务器能够使用多种字符集，而且在服务器、数据库、数据表、数据列以及字符串常数多个级别上设置不同的字符集。

1.4.1. MySQL4.1以前版本

MySQL4.1以前版本的字符集由服务器默认指定，默认值是编译系统时指定的字符集，该字符集也可通过在启动服务器时指定--default-character-set来修改。这种修改会对数据表的索引造成影响，因为索引的顺序是和字符集有关的，修改字符集会使这个已排序的顺序产生错误。要解决该问题，我们要用修改后的字符集的排序顺序重建表的索引。重建索引有以下几种方法：

- 用mysqldump导出数据，再清除表里的内容，最后用导出文件重新导入。数据表的索引将在导入数时重建。该方法适用于所有数据表类型。
- 删除索引，然后重建。用alter table命令或drop index和create index命令来完成。该方法也适用于所有数据表类型。但该方法需要我们了解重建索引的精确定义。

- MyISAM数据表的索引可以用myisamchk程序的--recover和--quick选项加上一个用来设定新字符集的--set-character-set选项进行重建。还可以用mysqlcheck程序的--repair和--quick选项或者一个带QUICK选项的REPLACE TABLE语句来重建索引，这种方式较方便。

1.4.2. MySQL4.1以后版本

MySQL4.1以后的版本对字符集的支持好了很多，具有以下新增功能：

- 支持服务器同时使用多种字符集。
- 允许在服务器，数据库，数据表，数据列等多级别上设置不同的字符集。
    - 服务器的默认字符集在编译时选定，但可在启动服务器时用--default-character-set选项来更改。
    - 用ALTER DATABASE db_name DEFAULT CHARACTER SET charset来设置数据库字符集。 如果只有default参数，则使用服务器的字符集。
    - 用CREATE TABLE table_name(...) CHARACTER SET = charset设置数据表字符集。如果charset为default，则使用数据表所在数据库的字符集作为数据表的字符集。
    - 在数据列中，可用CHARACTER SET charset属性来设置数据列的字符集。charset不能是default，如果没有该属性，则默认使用数据表的字符集。允许设置字符集的数据列有char，varchar(不带binary属性)及TEXT类型。
    - 用_charset str转换字符串常数的字符集。如：_utf8 'mysql'，_latinl 'oracle'。该方法只适用于括在引号内的字符串，其它十六进制常数 、字符串表达式等可用CONVERT()函数进行转换，如：SELECT CONVERT(str USING charset)。
- 通过MySQL提供的函数可进行字符集转换和查询。
- 新增的COLLATE操作符使我们可按某一种字符集的排序顺序来处理另一种字符集的数据。如：SELECT a from t ORDER BY a COLLATE utf-8；
- 用SHOW CHARACTER SET命令可显示服务器支持的字符集列表。
- 当服务器转换到另一种字符集时，会自动对索引进行重新排序。
- 通过UTF-8和UCS2字符集提供了Unicode支持。

MySQL现在还不支持：1,在同一个字符串里混用不同字符集的字符；2,在同一个数据列里混用不同的字符集。

1.4.3. 各级字符集的查询方法

- 服务器级

```
SHOW CHARACTER SET；可查出可供使用的所有字符集。
SHOW VARIABLES LIKE 'character_set'；可查出服务器的默认字符集。
```

- 可查出数据库级的字符集。

```
SHOW CREATE DATABASE db_name；
```

- 两条命令可查出数据表的字符集。

```
SHOW CREATE TABLE table_name；
SHOW TABLE STATUS LIKE 'table_name'
```

- 以下几命令可查出数据列的字符集：

```
DESCRIBE table_name;
SHOW COLUMNS FROM table_name;
SHOW CREATE TABLE table_name;
```

- 用CHARSET()函数可确定特定字符串，字符串表达式或数据列值相关联的字符串的字符集。如：SELECT CHARSET(str)。

1.4.4. Unicode支持

MySQL提供两种字符集来支持Unicode。一个是UTF-8，一种可变长的编码格式，需用1至4个字节来表示一个字符；另一个是UCS2，该字符集中的每个字符需要用两个字节来表示。

1.5. 如何选择数据列类型？

选择正确的数据列类型能大大提高数据库的性能和使数据库具有高扩展性。在选择数据列类型时，请从以下几个方面考虑：

- 存放到数据列中的数据类型。
- 数据值的取值范围。
- 考虑性能和处理效率。
  - 数值操作比字符操作快。
  - 小类型的处理速度比大类型快。
  - 不同数据表中固定长度类型和可变长度类型的处理效率是不同的。

可变长度类型在经过删除和修改操作后容易产生碎片，降低系统性能，需定期运行OPTIMIZE TABLE命令以优化数据表。

固定长度类型由于有固定的长度，所以容易确定每条记录的起始点，可加快数据表的修复速度。

在MyISAM和ISAM表中使用固定长度类型数据列有助改善数据库性能。

在InnoDB表中，固定长度和可变长度数据列类型都以相同方式存储，所以固定长度数据列类型并没有性能优势，反而由于可度长度数据列类型由于占用存储空间较少，所以处理速度会快些。

  - 可索引类型能加快数据的查询速度。
  - 明确指定数据列的NOT NULL属性可使MySQL在检索过程中不用去判断数据列是否是NULL，所以可加快处理速度。

- 数据如何进行比较，是否区分大小写。
- 是否要在数据列上建立索引。

1.6. 表达式操作符

Table 1.5. 算术操作符

操作符	语法	含义
+	a + b	相加
-	a - b	相减
-	- a	求负
*	a * b	乘法
/	a / b	除法
%	a % b	求余

Table 1.6. 逻辑操作符

操作符	语法	含义
AND 或 &&	a AND b 或 a && b	逻辑与，若两个操作数同时为真，则为真
OR 或 ||	a OR b 或 a || b	逻辑或，只要有一个操作数为真，则为真
XOR	a XOR b	逻辑异或，若有且仅有一个操作数为真，则为真
NOT 或 ！	NOT a 或 !a	逻辑非，若操作数为假，则为真

Table 1.7. 位操作符

操作符	语法	含义
&	a & b	按位与，若操作数同位同为1,则该位为1
|	a | b	按位或，若操作数同位有一位为1,则该位为1
^	a ^ b	按拉异或，若操作数同一位分别为1和0,则该位为1
<<	a << b	把a中的各个位左移b个位置
>>	a >> b	把a中的各个位右移b个位置

Table 1.8. 比较操作符

操作符	语法	含义
=	a = b	若两个操作数相等，则为真
<=>	a <=> b	若两个操作数相等，则为真，可用于NULL值比较
!= 或 <>	a != b 或 a <> b	若两个操用数不等，则为真
<	a < b	若a小于b，则为真
<=	a <= b	若a小于或等于b，则为真
>	a > b	若a大于b，则为真
>=	a > b	若a大于或等于b，则为真
IN	a IN (b1,b2,...)	若a等于b1,b2,...中的某一个，则为真
BETWEEN	a BETWEEN b AND c	若a在b和c之间(包括b和c)，则为真
NOT BETWEEN	a NOT BETWEEN b AND c	若a不在b和c之间(包括b和c)，则为真
LIKE	a LIKE b	SQL模式匹配，若a匹配b，则为真
NOT LIKE	a NOT LIKE b	SQL模式匹配，若a不匹配b，则为真
REGEXP	a REGEXP b	正则表达式匹配，若a匹配b，则为真
NOT REGEXP	a NOT REGEXP b	正则表达式匹配，若a不匹配b，则为真
IS NULL	a IS NULL	若a为NULL，则为真
IS NOT NULL	a IS NOT NULL	若a不为NULL，则为真

> LIKE模式匹配中的“%”匹配任意个字符，“_”匹配一个字符。匹配不区分大小写字符。

Table 1.9. 操作符优先级(由高至低排列)

<table>
<tr><th>操作符</th></tr>
<tr><td>BINARY，COLLATE</td></tr>
<tr><td>NOT、！</td></tr>
<tr><td>^</td></tr>
<tr><td>XOR</td></tr>
<tr><td>-(一元求负操作符)、~(一元取反操作符)</td></tr>
<tr><td>*、/、%</td></tr>
<tr><td>+、-</td></tr>
<tr><td><<、>></td></tr>
<tr><td>&</td></tr>
<tr><td>|</td></tr>
<tr><td><、<=、=、<=>、!=、<>、>=、>、IN、IS、LIKE、REGEXP、RLIKE</td></tr>
<tr><td>BETWEEN、CASE、WHEN、THEN、ELSE</td></tr>
<tr><td>AND、&&</td></tr>
<tr><td>OR、||</td></tr>
<tr><td>:=</td></tr>
</table>

1.7. 类型转换

在MySQL的表达式中，如果某个数据值的类型与上下文所要求的类型不相符，MySQL则会根据将要进行的操作自动地对数据值进行类型转换。如：

```
1 + '2'      会转换成1 + 2 = 3
1+ 'abc'     会转换成1 + 0 = 1 由于abc不能转换成任何的值，所以默认为0
```

MySQL会根据表达式上下文的要求，把字符串和数值自动转换为日期和时间值

对于超范围或非法的值，MySQL也会进行转换，但转换出来的结果是错误的。出现该情况时，MySQL会提示警告信息，我们可捕获该信息以进行相应的处理。

Chapter 2. 查询优化

数据库是数据的集合，与数学的集合论有密不可分的关系。

为提高查询速度，我们可以：

- 对数据表添加索引，以加快搜索速度；

- 通过编程技巧最大限度地利用索引；
- 优化查询语句，以使服务器最快响应多客户的请求。
- 研究硬件处理过程，减少物理约束。

2.1. 索引

索引技术是关系数据查询中最重要的技术。如果要加提升数据库的性能，索引优化是首先应该考虑的。因为它能使我们的数据库得到最大性能方面的提升。

索引的优点：

- 没有索引的表是没有排序的数据集合，如果要查询数据需进行全表扫描。有索引的表是一个在索引列上排序了数据表，可通过索引快速定位记录。在MyISAM和ISAM数据表中，数据行保存在数据文件中，索引保存在索引文件中。BDB与InnoDB数据表把数据与索引放在同一个文件中。
- 在多表关联查询中，索引的作用就更大。如果没有索引，在最坏的情况下，全表扫描的次数可能是各表数据行的组合个数，可能是一个天文数字。这样的查询是破坏性的，可能会造成数据库瘫痪。
- 对于使用了MIN()或是MAX()函数的查询，如果相关的数据列上有索引，MySQL能直接找到该最大、最小值的行，根本不用一个一个地去检查数据行。
- 索引加快ORDER BY 和 GROUP BY子句的操作。
- 当在数值型数据列上查询数据，而该列有索引，索引能使MySQL根本不用去读取数据行，直接从索引取值。

索引的缺点：

- 索引需占用磁盘空间。
- 索引会减慢在索引数据列上的插入、删除和修改操作。

索引列的选择

- 索引应该创建在搜索、排序、分组等操作所涉及的数据列上。也就是说，在where子句，关联检索中的from子句、order by或group by子句中出现过的数据列最适合用来创建索引。
- 尽量使用唯一索引，它能使索引发挥最好的效能。
- 尽量用比较短的值进行索引。当对字符串进行索引时，应该指定一个前缀长度，比如对字符串的前10位或20位的字符进行排序，而不用把整个字符串几十个字符用来索引排序。这样能减少磁盘I/O，提高处理速度。最重要的一点是，键值越短，索引缓冲区里容

纳的键值也就越多，而MySQL同时保存在内存里的索引越多，索引缓冲区的命中率也就越高。当然，只对数据列第一个字符进行索引是没什么意义的。

- 充分利用最左前缀。所谓最左前缀也就是在复合索引中最边的索引列。如复合索引(a,b,c)，其中a就是最左前缀。它是使用率最高的索引，需认真选择。
- 不要建太多索引，索引是会消耗系统资源的，要适可而止。
- 索引主要用于<、<=、=、>=、>、BETWEEN等的比较操作中，所以索引应该建立在与这样操作相关的数据列上。
- 利用慢查询日志来找出性能差的查询，通过mysqldumpslow可查看该日志。针对性能差的查询可利用索引来加快查询速度。

2.2. 查询优化程序

当我们发一条查询命令时，MySQL分对它进行分析，以优化查询。把explain语名放到查询前面可显示查询的执行路线，对优化查询提供有用的信息。以下几个原则可帮助系统挑选和使用索引：

- 尽量对同类型的数据列进行比较。如：VARCHAR(5)和VARCHAR(5)是同类型的，CHAR(5)和VARCHAR(5)是不同类型的。
- 尽量让索引的数据列在比较表达式中单独出现，不要把它包含在函数或复杂表达式。否则索引会不起作用。
- 尽量不要在LIKE模式的开头使用通配符。如：%string%。
- 对于MyISAM和BDB数据表，用ANALYZE TABLE语句让服务器对索引键值的分布进行分析，为优化程序提供更有价值的信息。另一个方法是用myisamchk --analyze(适用于MyISAM表)或isamchk --analyze(适用于ISAM表)命令。
- 用EXPLAIN语句来分析查询语句的执行效率。检查查询所使用的索引是不是能够迅速地排除不符合条件的数据行，如果不是，可以试着用STRAIGHT_JOIN强制各有关数据表按指定顺序进行关联。
- 尝试查询的不同写法，比较运行情况。
- 不要滥用MySQL的类型自动转换功能。自动转换会减慢查询的速度并会使有关的索引失效。

2.3. 数据列类型与查询效率

选用适当的数据列类型有助于提高查询命令的执行速度，下面是几点关于如何选择合适数据列类型的建议：

- 尽量选用尺寸较小的数据列。这样能节约磁盘空间和加快查询速度。如果较短的数据列上建有索引，则索引的处理速度会进一步提高。

- 针对数据列类型，尽量选择最适用的数据表类型。如固定长度数据列在MyISAM或ISAM数据表中的速度是最快的，所以在这样数据表中尽量使用char类型而不是varchar类型来保存字符串数据。对于InnoDB数据表类型，由于varchar类型可有效减少占用空间，从而减少磁盘I/O，所以使用varchar类型是有利的。对于BDB类型数据表，使用定长和不定长列类型的区别就不大，可任选一种。

- 尽量把数据列声明为NOT NULL，以节约存储空间和加快处理速度。

- 对于取值范围有限的数据列，考虑使用ENUM数据列类型。ENUM数据列类型在MySQL中的处理速度是很快。

- 使用PROCEDURE ANALYSE()语句来分析数据表，它会对数据列的声明提出建议，我们可根据建议进行修改。

```
select * from table_name PROCEDURE ANALYSE();
select * from table_name PROCEDURE ANALYSE(16,256);    #(16,256)含义是：如果某列的不同取
```

- 用OPTIMIZE TABLE语句对容易出现碎片的数据表进行整理。包含可变长数据列的数据表都会产生碎片，从而占用多余的磁盘空间和影响查询速度。所以要定期运行OPTIMIZE TABLE语句以防止数据表查询性能降低。但该语句只对MyISAM数据表有效。对各种数据表通用的碎片整理方法是这样的：先用工具程序mysqldump导出数据表，再删除数据表后重建，如：

```
$ mysqldump --opt db_name table_name &gt; dump.sql
$ mysql db_name &lt; dump.sql
```

- 把非结构化和变化大的数据放在BLOB数据列里，定期用OPTIMIZE TABLE命令优化。

- 人为地给数据表增加一个数据列，以充当索引。做法是这样的，先根据数据表里的其它数据列计算出一个散列值，并保存在一个数据列里，然后通过搜索散列值来检索数据行。注意，该技巧只适用于精确匹配型查询。散列值在大于，小于等的操作中不起作用。散列值可以MD5()(适用于3.23及以上版本)，SHA1()(适用于4.0.1及以上版本)，CRC32()(适用于4.1及以上版本)等函数生成。使用散列值支检索BLOB和TEXT值的做法比直接检索BLOB和TEXT本身的做法快。

- 尽量避免对大尺寸的BLOB值进行检索。如果要检索都应该通过它的上面提到散列值先进行筛选。而不应该盲目地在网络中传送大量BLOB值。

- 如果把BLOB值剥离到另外一个数据表里去，可实现数据表中其它数据列转变成固定长度数据列的话。就即可减少数据表碎片，又可使在原始表中的select *查询不会把大尺寸的BLOB值不必要地通过网络传送。

2.4. 有效地加载数据

有时我们需大量地把数据加载到数据表，采用批量加载的方式比一个一个记录加载效率高，因为MySQL不用每加载一条记录就刷新一次索引。下面介绍几个有助于加快数据加载的操作：

- 使用LOAD DATA语句要比INSERT语句的加载速度快。
- LOAD DATA比LOAD DATA LOCAL语句的效率高。前者可由服务器直接从本地磁盘读取加载数据，后者需由客户程序去读取文件并通过网络传送到服务器。
- 如果一定要用INSERT语句，应尽量在一条语句中插入多个数据行。
- 如果必须使用多条INSERT语句，则应尽量把它们集中在一起放到一个事务中进行处理，而不是在自动提交模式下执行它们：如：

  ```
  BEGIN;
  INSERT INTO table_name values (...);
  INSERT INTO table_name values (...);
  INSERT INTO table_name values (...);
  ...
  COMMIT;
  ```

  对于不支持事务的表，应对表进行写锁定，然后在表锁定期间对表进行INSERT操作，如：

  ```
  LOCK TABLES table_name WRITE;
  INSERT INTO table_name ...;
  INSERT INTO table_name ...;
  INSERT INTO table_name ...;
  ...
  UNLOCK TABLES;
  ```

- 利用客户/服务器通信协议中的压缩功能以减少网络传输的数据量。但该压缩会消耗大量的系统资源，所以小心使用。
- 尽量让MySQL插入默认值。不要在INSERT中写太多值，以减少网络传输量和服务器端的语法分析时间。
- 对于MyISAM和ISAM数据表，如果需加载大量数据，应先建立一个没索引的表，加载数据后再创建索引。该方法不适用于InnoDB或BDB数据表。

禁用和重新激活索引的方法有两种：

- 使用ALTER TABLE语句的DISABLE KEYS和ENABLE KEYS命令，如：

  ```
  ALTER TABLE table_name DISABLE KEYS;
  ALTER TABLE table_name ENABLE KEYS;
  ```

- 使用myisamchk或isamchk工具。如：

```
$ myisamchk --keys-used=0 table_name                 #禁止
$ myisamchk --recover --quick --key-used=n table_name  #激活
```

n是用来表明需要激活索引的位掩码，第0位对应第一个索引，如果有三个索引，n值就是7(二进制111)。索引编号可以下命令确定：

```
$ myisamchk --description table_name
```

2.5. 调度和锁定

在很多客户一起查询数据表时，如果使客户能最快地查询到数据就是调度和锁定做的工作了。在MySQL中，我们把select操作叫做读，把对数据表修改增加的操作(INSERT,UPDATE,REPLACE...)叫做写。MySQL的基本调度策略可以归纳为以下两条：

- 写入请求将按它们到达服务器的顺序进行处理；
- 写操作的优先级要高于读操作。

MyISAM和ISAM数据表的调度策略是在数据表锁的帮助下实现的，在客户程序要访问数据表之前，需获得相应的锁，在完成对数据表的操作后，再释放该锁。锁的管理通常由服务器管理，也可人为地用LOCK TABLES和UNLOCK TABLES命令来申请和释放锁。写操作时，需要申请一个独占性的锁，也就是说在写操作其间，该表只能由写操作的客户使用。读操作时，客户必须申请一个允许其他客户对数据表进行写操作的锁，以确保客户在读的过程中数据表不会发生改变。但读操作锁不是独占的，可有多个读操作同时作用于同一个数据表。

通过一些修饰符可影响调度策略，如LOW_PRIORITY(用于DELETE,INSERT,LOAD DATA,REPLACE,UPDATE语句)、HIGH_PRIORITY(用于SELECT语句)、DELAYED(用于INSERT和REPLACE语句)。它们的作用是这样的：

- LOW_PRIORITY会使写操作的优先级降低到读操作以下，也就是说读操作会阻塞该级别的写操作，SELECT的HIGH_PRIORITY有类似的作用。
- INSERT语句中的DELAYED修饰会使插入操作被放入一个“延迟插入”队列。并返回状态信息给客户，使客户程序可在新数据行还没插入到数据表之前继续执行后面的操作。如果一直有客户读该数据表，新数据行会一直待在队列中，直到数据表没有读操作时，服务器才会把队列中的数据行真正插入到数据表中。该语句可用在以下场合，在一个有冗长查询的数据表中插入数据，而你又不想被阻塞，你就可发出INSERT DELAYED语句，把插入操作放入服务器“延迟插入”队列，你无需等待就马上可进行接下来的操作。

- 当一个数据表里从未进行过删除操作或刚刚对它进行过碎片整理的情况下，用INSERT语句插入的数据行只会被添加到数据表的末尾，而不会插入到数据表的中间位置。这样，对于MyISAM表，MySQL允许在有其它客户正在读操作的时间进行写操作。我们称之这并发插入。要使用该技巧，需注意以下两个问题：
    - 不要在INSERT语句中使用LOW_PRIORITY修饰符。
    - 读操作应用LOCK TABLES ... READ LOCAL而不是用LOCK TABLES ... READ语句来进行数据表读锁定。LOCAL关键字只对数据表中已存在行进行锁定，不会阻塞把新行添加到数据表末尾。

BDB数据表使用页面级操作锁，InnoDB数据表使用数据行级操作锁。所以这两种表的并发性比MyISAM和ISAM数据表这种表级锁的并发性会好很多。其中InnoDB的并发性最好。综上所述，我们可得出以下结论：

- MyISAM和ISAM数据表的检索速度最快，但如果在检索和修改操作较多的场合，会出锁竞争的问题，造成等待时间延长。
- BDB和InnoDB数据表能在有大量修改操作的环境下提供很好的并发性，从而提供更好的性能。
- MyISAM和ISAM数据表由于进行表级锁定，所以不会出现死锁现象，BDB和InnoDB数据表则存在死锁的可能性。

2.6. 服务器优化

优化原则：

- 内存里的数据要比磁盘上的数据访问起来快；
- 站数据尽可能长时间地留在内存里能减少磁盘读写活动的工作量；
- 让索引信息留在内存里要比让数据记录的内容留在内存里更重要。

针对以上几个原则，我们应该调整服务器：

- 增加服务器的缓存区容量，以便数据在内存在停留的时间长一点，以减少磁盘I/0。下面介绍几个重要的缓冲区：
    - 数据表缓冲区存放着与打开的数据表相的信息，它的大小可由服务器参数“table_cache”设置。Opened_tables参数记录服务器进行过多少次数据表打开操作，如果该值变化很大，就可能是数据表缓冲区已满，需把一些不常用的表移出缓冲区，以腾出空打开新的数据表。可用以下命令查看Opened_tables的值：

      ```
      SHOW STATUS LIKE 'Opened_tables';
      ```

   - 在MyISAM和ISAM数据表中，索引被缓存在“key buffer”里，它的大小由服务器参数“key_buffer_size”来控制。系统默认的大小是8M，如果内存充足的话可适当扩大该值，以使更多索引块缓存在该区里，以加快索引的速度。
   - InnoDB和BDB数据表也各有一个缓冲区，分别叫innodb_buffer_pool_size和bdb_cache_size。InnoDB还有一个日志缓冲区叫innodb_log_buffer_size。
   - 自4.0.1开始，MySQL多了一个缓冲区，叫查询缓冲区，主要用来存放重复执行的查询文本和结果，当再次遇到相同的查询，服务器会直接从缓冲区中返回结果。该功能是内建的功能，如不想支持该功能，可在编译服务器时用configure脚本的--without-query-cache选项去掉该功能。

     查询缓冲区由三个服务器参数控制，分别是：

     1、query_cache_size
     控制缓冲区的大小，如果该值为0,则禁用查询缓冲功能。设置方法是在选项文件中设置:

     ```
     [mysqld]
     set-variable = query_cache_size = 16M
     ```

     这样就设置了一个16M的查询缓冲区

     2、query_cache_limit 缓冲结果集的最大容量(以字节为单位)，如果查询的结果集大于该值，则不缓冲该值。

     3、query_cache_type

     缓冲区的操作模式。

     0表示不进行缓冲；

     1表示除SELECT SQL_NO_CACHE开头的查询外，其余的都缓冲；

     2表示只对以SELECT SQL_ON_CACHE开头的查询进行缓冲。

     默认情况下，按服务器的设置进行缓冲，但客户端也可通过命令改变服务器设置。客户端可直接用SELECT SQL_NO_CACHE和SELECT SQL_CACHE命令来要求服务器缓冲或不缓冲查询结果。如果不想每条查询都写参数，我们也可在客户端用SET SQL_QUERY_CACHE_TYPE = val;来改变服务器的查询缓冲行为。val可取值0，1,2或OFF，ON，或DEMAND。

- 禁用用不着的数据表处理程序。如服务器是从源码创建，就可彻底禁用ISAM，InnoDB和BDB数据表。
- 权限表里的权限关系应尽可能简单，当然了，是要在保证安全的前提下。

- 在从源码创建服务器时，尽量使用静态库而不是共享库来完成其配置工作。静态库的执行速度更快，但如果要加载用户定义函数(UDF)的话，就不能使用静态库，因为UDF机制必须依赖动态库才能实现。

2.7. 硬件优化

为了提高数据运行速度，升级硬件是最直接的解决方案。针对数据库应用的特点，在升级硬件时应考虑以下内容：

- 对于数据库服务器，内存是最重要的一个影响性能因素。通过加大内存，数据库服务器可把更多的数据保存在缓冲区，可大大减少磁盘I/O，从而提升数据库的整体性能。
- 配置高速磁盘系统，以减少读盘的等待时间，提高响应速度。
- 合理分布磁盘I/O，应把磁盘I/O分散在多个设备上，以减少资源竞争，提高并行操作能力。
- 配置多处理器，MySQL是多线程的数据库，多处理器可同时执行多个线程。

Chapter 3. 数据库管理

数据库是一个复杂而又关键的系统，为确保系统安全、高效运行，需熟悉数据库内部的运作机制，掌握各种维护工具，并做好日常的管理工作。下面列举几项主要工作职责：

- 服务器的关闭和启动；
- 管理用户帐号；
- 管理日志文件；
- 数据库备份恢复；
- 数据库优化；
- 确保数据库数据安全；
- 数据库软件升级。

3.1. 数据目录

数据目录是用来存放数据表和相关信息的地方，是数据库的核心。在MySQL中的数据目录根据不同平台的有一些差异：

- 在UNIX/Linux系统上，如果用源码编译安装，数据目录的位置默认是在/usr/local/mysql/var中；

- 在UNIX/Linux系统上，如果用二进制发行版安装，数据目录的位置默认是在/usr/local/mysql/data中；
- 在WINDOWS系统上，数据目录的位置默认是在c:/mysql/data中；

> 在服务器启动时，可用--datadir=dir_name来指定数据目录，也可把它写到配置文件中。

我们还可用命令向服务器查询数据目录的位置，数据目录的变量名是datadir，如：

```
% mysqladmin variables
```

如果在一台机器上同时运行多个服务器，则可根据端口的不时来查询每个服务器的数据目录，如：

```
% mysqladmin --host=127.0.0.1 --port=port_number variables
```

如果--host是localhost，系统则会用一个UNIX套接字去连接数据库服务器，这时要使用--socket选项，所以查询语句变成：

```
% mysqladmin --host=localhost --socket=/path/to/socket variables
```

```
mysql> SHOW VARIABLES LIKE 'datadir';
```

- 在windows NT平台上可以使用“.”作为一条命名管道连接的主机名，用--socket选项给出命名管道的名字，如：

```
c:\ mysqladmin --host=. --socket=pipe_name variables
```

- 配置文件的中[mysqld]段中的datadir=/path/to/datadir设置也可查询到数据目录。
- 在mysqld程序的帮助信息里也有程序编译时默认的数据目录信息，可用以下命令显示：

```
% mysqld --help
```

数据目录是存放数据文件的地方，每个数据库对应目录的不同文件。InnoDB数据表由于用表空间来管理数据库，所以就没这种对应关系。但也是保存在数据目录中的，在数据目录除保存数据库文件外，还可能会保存以下几类文件：

- 服务器的配置文件，my.cnf；
- 服务器的进程ID(PID)文件；
- 服务器的日志文件和状态文件，这些文件对管理数据库有重要的价值；

- DES密钥文件或服务器的SSL证书与密钥文件。

数据目录中的所有数据库全部由服务器(mysqld)来管理，客户端不直接操作数据。服务器是客户使用数据的唯一通道。

在MySQL中，每个数据库其实就是在数据目录下一个子目录，show databases命令相当于列出数据目录中的目录清单。create database db_name命令会在数据目录下新建一个db_name的目录，以存放数据库的数据文件。所以我们也可下面的shell命令方式来建立一个空数据库：

```
% cd datadir
% mkdir db_name
% chmod u=rwx,go-rwx db_name
```

同理，删除数据库drop database db_name也就是删除数据目录中一个名为db_name的目录及目录中的数据表文件。我们也可用shell这进行操作：

```
% cd datadir
% rm -rf db_name
```

> 比较shell方式与drop database方式，drop database db_name命令不能删除db_name目录中创建的其它非数据表文件；由于InnoDB是表空间来管理数据表，所以不能用rm或del命令删除InnoDB的数据表。

3.2. MySQL数据表在系统中表现形式

MySQL数据表类型有：ISAM、MyISAM、MERGE、BDB、InnoDB和HEAP。每种数据表在文件系统中都有不同的表示方式，有一个共同点就是每种数据表至少有一个存放数据表结构定义的.frm文件。下面介绍每种数据表文件：

- ISAM数据表是最原始的数据表，有三个文件，分别是：

  .frm，存放数据表的结构定义；

  .ISD，数据文件，存放数据表中的各个数据行的内空；

  .ISM，索引文件，存放数据表的所有索引信息。

- MyISAM数据表是ISAM数据表的继承者，也有三个文件，分别是：

  .frm，结构定义文件；

  .MYD，数据文件；

  .MYI，索引文件。

- MERGE数据表是一个逻辑结构，代表一组结构完全相同的MyISAM数据表构成的集合。它在文件系统中有二个文件，分别是：

  .frm，结构定义文件；

  .MRG，构成MERGE表的MyISAM数据表清单，每个MyISAM数据表名占一行。也就是说可通过改变该表的内容来改变MERGE数据表的结构。修改前请先刷新缓存(flush tables)，但不建议这样修改MERGE数据表。

- BDB数据表用两个文件来表示，分别是：

  .frm，结构定义文件；

  .db，数据表数据和索引文件

- InnoDB由于采用表空间的概念来管理数据表，所以它只有一个与数据表对应.frm文件，同一目录下的其它文件表示为表空间，存储数据表的数据和索引。

- HEAP数据表是一个存在于内存中的表，所以它的数据和索引都存在于内存中，文件系统中只有一个.frm文件，以定义结构。

了解MySQL数据表在文件系统中表现形式后，我们可知道，创建、修改或删除数据表，其实就是对这些文件进行操作。例如一些数据表(除InnoDB和HEAP数据表外)，我们可直接在文件系统中删除相应的文件来删除数据表。

```
% cd datadir
% rm -f mydb/mydb.*
```

以上命令可删除mydb数据库中的mydb数据表。

3.3. 数据表最大尺寸限制

在MySQL中影响数据表尺寸的因素有很多，下面分别进行介绍：

- MySQL数据表类型的不同对数据表尺寸的限制：
  - ISAM数据表中单个.ISD和.ISM文件的最大尺寸为4G；
  - MyISAM数据表中单个.MYD和.MYI文件的默认最大尺寸也是4G，但可在创建数据表时用AVG_ROW_LENGTH和MAX_ROWS选项把这个最值扩大到800万TB。
  - MERGE数据表的最大尺寸是它的各组成MyISAM数据表的最大尺寸之和。
  - BDB数据表的尺寸受限于BDB处理程序所允许的.db文件的最大尺寸。这个最大尺寸随着数据表页面尺寸(编译时确定)而变化，但即使是最小的页面尺寸(512字节)，.db文件的最大尺寸也可达2TB。

    - InnoDB数据表的表空间的最大尺寸是40亿个页面，默认的页面尺寸是16K，该值可在8K到64K之间，在编译时确定。InnoDB数据表的最大尺寸也就是表空间的最大尺寸。

- 操作系统对文件的尺寸限制，一般文件系统都对单个文件不得超过2G的限制。该约束会对数据库文件造成限制。InnoDB数据表可通过利用未格式化硬盘作为表空间来绕过该限制。
- 对于数据和索引分开两个文件存放的数据表，其中任何一个文件达到操作系统文件的最大限制，数据库表也就达到最大尺寸。
- 包含AUTO_INCREMENT数据列的表受到该数据列类型最大上限值的限制。
- 由于InnoDB数据表用表空间来管理，一个表空间可同时空纳多个数据表，所以数据表的最大尺寸受系统文件和同一表空间中数据表空间的约束。

3.4. 状态文件和日志文件

在MySQL数据目录中还包含着许多状态文件和日志文件，这些文件的文件名都是以主机名加上相关后缀来命名的。下面是这些文件的一个说明列表：

Table 3.1. 状态文件和日志文件

文件类型	默认名	文件内容
进程ID文件	hostname.pid	MySQL服务器进程的ID
常规查询日志	hostname.log	连接/断开连接事件和查询信息
慢查询日志	hostname-slow.log	记录查询时间很长的命令信息
变更日志	hostname.nnn	创建或修改数据表结构和内容的查询命令信息
二进制变更日志	hostname-bin.nnn	创建或修改数据表结构和内容的查询命令的二进制表示法
二进制变更日志的索引文件	hostname-bin.index	使用中的“二进制变更日志”列表
错误日志	hostname.err	记录“启动/关闭”事件和异常情况

变更日志和二进制变更日志主要用于MySQL数据库服务器的崩溃恢复中，由于变更日志记录了数据库的所有变更操作，所以可以进行事件重放。具体操作请参考相关数据库备份恢复章节。对于变更日志，我们可用--log-long-format选项来让它以扩展方式记录有关事件。扩展方式可记录谁发出查询和什么时候发出查询的信息。可使我们更好地掌握客户端的操作情况。日志记录着查询命令的所有操作，里面可能会有一些敏感信息。所以我们要确保日志文件的安全。

3.5. 调整MySQL数据目录位置

MySQL数据库的数据目录位置，包括目录里的各种文件的位置)可根据实际情况进行调整。调整的方法有两种，一种是使用符号链接；一种用服务器启动选项。下面一个列表说明了数据目录及目录中文件各自适宜采用的方法：

Table 3.2. MySQL数据目录及目录中文件位置的调整方法

调整对象	适用方法
整个数据目录	启动选项和符号链接
数据库目录	符号链接
数据表	符号链接
InnoDB数据表空间	启动选项
PID文件	启动选项
日志文件	启动选项

下面是各种调整方法的具体操作过程：

- 在调整MySQL的数据目录时，要先停止服务器，再把数据目录移动到新的位置。接着，我们可选择在原来目录下创建一个符号链接指向新的位置，或者用启动选择--datadir指向新的数据目录。推荐用创建符号链接的方法，因为如果那个数据目录中有my.cnf文件，相应的服务器还能找到它。
- 数据库只能存在于MySQL数据目录中，所以只能使用符号链接的方法调整它的位置。在Linux系统的操作步骤如：

  1. 关闭服务器；
  2. 把数据库目录拷贝到新的位置；
  3. 删除原来的数据库目录；
  4. 在原来的MySQL数据目录中创建一个同名符号链接指向新的位置；
  5. 重新启动服务器。

  在windows下的操作方法不些不同，操作方法如下：

  1. 关闭服务器；
  2. 把数据库目录移动新的位置；
  3. 删除原来的数据库目录；

4. 在原来数据目录下建一个同名的.sym文件，在文件中输入数据库新目录的全路径，如c:\mysql\newdir\mydb。这个文件就相当于Linux下的符号链接；
5. 重启服务器。

> 为了支持符号链接功能，必须用--use-symbolic-links选项启动服务器；或在选项文件的[mysqld]节中添加use-symbolic-links选项。
>
> MySQL必须是3.23.16以上版本且是max服务器(mysqld-max或mysqld-max-nt)。

- 要移动数据表，必须满足以下所有条件才行：
  - MySQL的版本必须是4.0或以上的版本；
  - 操作系统必须有一个可用的realpath()调用；
  - 移动的数据表必须是MyISAM类型的数据表。

  在满足以上所有条件后，我们就可把.MYD数据文件和MYI索引文件移到新位置，再在原来位置创建两个同名符号链接指定新的位置。注意，.frm定义文件仍需留在原来的数据库目录中。

  如以上条件不能全部满足，最好不要移动数据表文件。否则一旦你运行ALTER TABLE、OPTIMIZE TABLE、REPAIR TABLE语句对移动过的数据表进行优化或修改，这样数据表就会回到原来的位置，使移动操作失效。因为这些命令的执行过程是这样的：它会先在数据目录中创建一个临时数据表并对这个临时数据表进行优化或修改，然后删除原来的数据表(这里是你为了移动数据表而创建的一个符号链接)，再把临时数据表更名为原来的数据表名称。这样一来，你移走的数据表就和这个数据库完全没有关系了。基于以下的不稳定因素，如无特殊必要，不建议移动数据表。

- InnoDB表空间是通过在选项文件中使用innodb_data_home_dir和innodb_data_file_path选项列出InnoDB表空间组成文件清单的方法来配置的，所以我们可通过修改这些选项来重新安置InnoDB表空间的组成文件。步骤如下：
  - 关闭服务器；
  - 移动组成表空间的文件；
  - 修改选项文件，指出组成表空间的文件的新位置；
  - 重启服务器。
- 状态文件和日志文件的位置可通过选项文件或启动服务器时指定。

Chapter 4. MySQL数据库日常管理

为了确保数据库平稳可靠运行，我们需进行维护和管理，这是每一位数据库管理员的职责。下面分几个专题分别介绍。

4.1. 数据库安全管理

MySQL数据库通过用户和密码来控制用户对数据库的访问，当我们新安装了一个数据库服务器时，MySQL的权限表设置是很不安全，它默认允许任何人不需要密码就可访问数据库。所以我们安装好服务器后第一件需要做的就是设置用户密码。

在MySQL中的mysql数据库的user数据表中存有用户的帐号信息，在初始状态下已存在root和一些匿名用户，且所有用户都没有设置密码。该数据表的这些用户信息是通过一个mysql_install_db脚本安装的。该表的主要列有：

- User，连接数据库的用户名。
- Host，允许连接到数据库服务器的主机名，“%”通配符代表所有主机。
- Password，连接密码，已加密。
- 其它权限列，以“Y”或“N”标识是否有效。

在这种状态下的数据库是极不安全的，我们可用以下命令轻易地访问数据库：

```
% mysql -h localhost -u root     #通过本地主机，root用户访问，不需要密码验证
% mysql -h localhost             #通过本地主机，匿名用户访问，不需要密码验证
```

设置MySQL用户帐号密码的方法有三种：

- 使用mysqladmin程序：

  ```
  % mysqladmin -h localhost -u root password "password"   #设置在本地以root身分登录的密码
  % mysqladmin -h remote -u root password "password"      #设置远程主机以root身分登录的密码
  ```

  在初始设置时，这两条语句都要运行，以确保数据库本地访问和网络访问的安全。

- 通过set password这条SQL语句设置：

  ```
  mysql> set password for 'root'@'localhost' = password('password');
  mysql> set password for 'root'@'remote' = password('password');
  ```

- 直接修改user权限表：

  ```
  mysql> use mysql;
  mysql> update user set password=password('password') where user='root';
  mysql> flush privileges;                          #重载权限表，使修改马上生效
  ```

MySQL使用驻留在内存中的权限表拷贝来进行访问控制，当使用mysqladmin和set password设置密码，MySQL会监察到权限表已被修改，它自动重载该表。而用update的方式，MySQL就监察不到变化，需手动用flush privileges命令刷新内存中的权限表，以使它马上生效。

为root用户设置密码后，如果需以root身份连接数据库，就需验证密码。我们可用以下语句连接数据库：

```
% mysql -u root -p
Enter password:                          #输入root的密码
```

在user表中，user列为空的为匿名用户。它也是没有密码的，我们需为它们设置一个密码，或干脆把它们删除。在windows系统上的本地匿名用户帐号和root用户有着同样的权限，这是一个很大的安全漏洞。应该把它删除或把权限削弱。

4.2. 服务器的启动和关闭

在Linux和windows平台下MySQL服务器的启动方式有很大不同，这里将分开介绍：

- Linux平台：

  Linux平台下，每一个进程都需由一个用户来运行，MySQL最好不要以root用户来运行。我们可创建一个mysql用户和mysql组，MySQL服务器程序目录和数据目录由这个用户和组所拥有，其它用户没有任何权限。以mysql用户来运行MySQL服务器。

  ```
  % mysqld --user=mysql     #即使以root用户执行该命令，MySQL数据库还是会与mysql用户ID关联。
  ```

  为了使服务器在系统启动时自动以mysql用户运行，需配置my.cnf配置文件 ，把user=mysql包含在[mysqld]段中。

  关闭服务器可用% mysql.server stop或% mysqladmin -u root -p shutdown

- windows平台：

  手动方式：直接运行c:\mysqld命令。

  作为服务方式：运行c:\mysqld-nt --install命令，把mysqld-nt安装为windows的服务，此后，每当windows启动时，它就会自动运行。mysqld-nt是一个支持命名管道的MySQL服务器。运行c:\mysqld-nt --remove可把服务删除。手动启动关闭服务的方法是运行c:\net start mysql和c:\net stop mysql命令。

4.3. 连接故障恢复

当由于误删mysql套接字时(/tmp/mysql.sock)，我们就不能通过套接字连接服务器。这时我们可通过tcp/ip来连接服务器，要建立一个tcp/ip连接，需用127.0.0.1代替locahost作为-h参数的值来连接服务器。如：

```
% mysqladmin -h 127.0.0.1 -u root -p shutdown       #关闭服务器再重启会重建套接字
```

当我们因为忘记root用户密码而不能连接服务器时，重设置密码的步骤如：

- 用 % kill -TERM PID关闭服务器，用-TERM信息可使服务器在关闭前把内存中的数据写入磁盘。如果服务器没有响应，我们可用% kill -9 PID来强制删除进程，但不建议这样做。这时内存中的数据不会写入磁盘，造成数据不完整。如果你是用mysql_safe脚本启动MySQL服务器的，这个脚本会监控服务器的运行情况并在它被终止时重启服务器，所以如需关闭服务器，要先终止该进程，然后再真正终止mysqld进程。
- 接着用--skip_grant-tables启动服务器。这时MySQL服务器将不使用权限表对连接操作进行验证。你就可在不提供root密码的情况下连接上服务器，并获得root的权限。这样你就可用上面介绍的修改密码的方法重设root用户的密码。注意：连接上服务器后，要马上执行flush privileges命令，使权限表读入内存并生效，以阻止其他的连接。该语句还重新激活grant语句，在MySQL服务器不使用权限表时，grant语句被禁用。
- 修改完root用户密码后，我们就可关闭服务器并重启使所有配置正常运作。

4.4. MySQL用户帐号管理

MySQL用户帐号管理主要用grant(授权)和revoke(撤权)两个SQL指令来管理。这两个指令实质是通过操作user(连接权限和全局权限)、db(数据库级权限)、tables_priv(数据表级权限)、columns_priv(数据列级权限)四个权限表来分配权限的。host权限表不受这两个指令影响。下面将会详细介绍用户权限管理的内容。

- GRANT语法说明：

```
GRANT privileges (columns)          #privileges表示授予的权限，columns表示作用的列(可选)
      ON what                       #设置权限级别，全局级、数据库级、数据表级和数据列级
      TO account                    #权限授予的用户，用"user_name"@"host_name"这种用户名、
      IDENTIFIED BY 'password'      #设置用户帐号密码
      REQUIRE encryption requirements       #设置经由SSL连接帐号
      WITH grant or resource management options;    #设置帐号的管理和资源(连接服务器次数或查
```

  示例：

```
mysql>grant all on db.* to 'test'@'localhost' identified by 'test';
```

  上例运行后的效果是，test用户只能通过‘test’密码从本机访问db数据库

```
mysql>grant all on db.* to 'test'@'%' identified by 'test';
```

上例运行后的效果是，test用户可通过‘test’密码从任意计算机上访问db数据库。‘%’代表任意字符，‘_’代表一个任意字符。主机名部份还可以是IP地址。

> 如果没有给定主机部份，则默认为任意主机，也就是'test'和'test'@'%'是等价的。

- **Table 4.1.** 访问权限表

```
| 权限 | 权限说明 |
| --- | --- |
| CREATE TEMPORARY TABLES | 创建临时数据表 |
| EXECUTE | 执行存储过程(暂不支持) |
| FILE | 操作系统文件 |
| GRANT OPTION | 可把本帐号的权限授予其它用户 |
| LOCK TABLES | 锁定指定数据表 |
| PROCESS | 查看运行着的线程信息 |
| RELOAD | 重新加载权限表或刷新日志及缓冲区 |
| REPLICATION CLIENT | 可查询主/从服务器主机名 |
| REPLICATION SLAVE | 运行一个镜像从服务器 |
| SHOW DATABASES | 可运行SHOW DATABASES指令 |
| SHUTDOWN | 关闭数据库服务器 |
| SUPER | 可用kill终止线程以及进行超级用户操作 |
| ALTER | 可修改表和索引的结构 |
| CREATE | 创建数据库和数据表 |
| DELETE | 删除数据表中的数据行 |
| DROP | 删除数据表和数据行 |
| INDEX | 建立或删除索引 |
| INSERT | 插入数据行 |
| REFERENCES | (暂时不支持) |
| SELECT | 查询数据行 |
| UPDATE | 更新数据行 |
| ALL | 所有权限，但不包括GRANT。 |
| USAGE | 无权限权限 |
```

- **Table 4.2.** 权限作用范围**(**由**ON**子句设置**)**

```
| 权限限定符 | 作用范围 |
| --- | --- |
| ON *.* | 全局级权限，作用于所有数据库 |
| ON * | 全局级权限，若未指定默认数据库，其作用范围是所有数据库，否则，其作用范围是当前数据库 |
| ON db_name.* | 数据库级权限，作用于指定数据库里的所有数据表 |
| ON db_name.tbl_name | 数据表级权限，作用于数据表里的所有数据列 |
| ON tbl_name | 数据表级权限，作用于默认数据库中指定的数据表里的所有数据列 |
```

- USAGE权限的用法：修改与权限无关的帐户项，如：

```
mysql>GRANT USAGE ON *.* TO account IDENTIFIED BY 'new_password';     #修改密码
mysql>GRANT USAGE ON *.* TO account REQUIRE SSL;                      #启用SSL连接
mysql>GRANT USAGE ON *.* TO account WITH MAX_CONNECTIONS_PER_HOUR 10; #设置资源
```

- 拥有WITH GRANT OPTION权限的用户可把自已所拥用的权限转授给其他用户，如：

```
mysql>GRANT ALL ON db.* TO 'test'@'%' IDENTIFIED BY 'password' WITH GRANT OPTION;
这样test用户就有权把该权限授予其他用户。
```

- 限制资源使用，如：

```
mysql>GRANT ALL ON db.* TO account IDENTIFIED BY 'password' WITH MAX_CONNECTIONS_PER_
```

允许account用户每小时最多连接20次服务器，每小时最多发出200条查询命令(其中更新命令最多为50条)

默认都是零值，即没有限制。FLUSH USER_RESOURCES和FLUSH PRIVILEGES可对资源限制计数器清零。

- REVOKE语法说明：

```
mysql>REVOKE privileges (columns) ON what FROM account;
```

示例：

```
mysql>REVOKE SELECT ON db.* FROM 'test'@'localhost';
删除test帐号从本机查询db数据库的权限
```

REVOKE可删除权限，但不能删除帐号，即使帐号已没有任何权限。所以user数据表里还会有该帐号的记录，要彻底删除帐号，需用DELETE命令删除user数据表的记录，如：

```
% mysql -u root -p
mysql>use mysql
mysql>DELETE FROM user where User='test' and Host='localhost';
mysql fulsh privileges;
```

REVOKE不能删除REQUIRE和资源占用的配置。他们是要用GRANT来删除的，如：

```
GRANT USAGE ON *.* TO account REQUIRE NONE;      #删除account帐号的SSL连接选项
GRANT USAGE ON *.* TO account WITH MAX_CONNECTIONS_PER_HOUR 0 MAX_QUERIES_PER_HOUR 0
```

4.5. 日志文件管理

有关MySQL的日志文件前面章节已简要讨论过了，主要有四种日志文件，分别是常规查询日志、慢查询日志、变更查询日志和二进制变更日志。这些日志文件的创建需在启动服务器时用选项指定。

Table 4.3. 日志启动选项

启动选项	激活日志
--log[=file_name]	常规日志文件
--log-bin[=file_name]	二进制变更日志文件
--log-bin-index=file_name	二进制变更日志文件索引文件
--log-update[=file_name]	变更日志文件
--log-slow-queries[=file_name]	慢查询日志文件
--log-isam[=file_name]	ISAM/MyISAM日志文件
--log-long-format	设置慢查询日志和变更日志的格式

BDB和InnoDB数据表的日志文件会自动创建不用指定选项。但可用以下选项指时日志文件的存放路径。

Table 4.4. BDB和InnoDB数据表日志选项

启动选项	用途
--bdb-logdir=dir_name	存放BDB日志文件的位置
--innodb-log_arch_dir=dir_name	存放InnoDB日志文件的归档目录
--innodb_log_group_home_dir=dir_name	存放InnoDB日志文件的位置

MySQL日志文件选项可在mysqld和mysqld_safe脚本中使用，也可在选项文件my.cnf的[mysqld]中使用。推荐在选项文件中使用，因为每次启动服务器的日志选项基本上都是一致的。

日志的刷新可用mysqladmin flush-logs命令或flush logs语句实现。另外，对MySQL服务器发送一条SIGHUP信号也会刷新日志。错误日志和DBD/InnoDB日志不能用以上方法刷新。

错误日志记录MySQL数据库系统的论断和出错信息，由mysqld_safe脚本创建，文件名默认为hostname.err，也可通过--err-log或选项文件的err-log语句指定另外的名字。如果直接用mysqld程序启动服务器，错误信息会直接输出到输出设备，也就是屏幕。但我们可用重定向方法把错误信息输出到其它地方，如把错误信息输出到/var/log/mysql.err文件中，可以执行以下语句：

```
% mysqld > /var/log/mysql.err 2>&1 &
```

在windows平台下，MySQL服务器默认把诊断信息写到数据目录的mysql.err文件中，并且不允许另外指定错误日志文件名。如在启动MySQL服务器时给出了--console选项，则MySQL会把诊断信息输出到控制台窗口而不创建错误日志。但如MySQL是作为一个服务运行，则--console选项不起作用。

4.5.1. 日志失效处理

在服务器正常运行中，会产生大量的日志文件。我们要对这些日志文件进行失效管理，以节省磁盘空间和方便查询。进行日志失效处理的方式主要有以下几种：

- 日志轮转。该方法适用于常规查询日志和慢查询日志这些文件名固定的日志文件，在日志轮转时，应进行日志刷新操作(mysqladmin flush-logs命令或flush logs语句)，以确保缓存在内存中的日志信息写入磁盘；

  日志轮转的操作过程是这样的：第一次轮转时，把log更名为log.1，然后服务器再创建一个新的log文件，在第二轮转时，再把log.1更名为log.2，把log更名为log.1，然后服务器再创建一个新的log文件。如此循环，创建一系列的日志文件。当到达日志轮转失效位置时，下次轮转就不再对它进行更名，直接把最后一个日志文件覆盖掉。例如：如果每天进行一次日志轮转并想保留最后7天的日志文件，就需要保留log.1--log.7共七个日志文件，等下次轮转时，用log.6覆盖原来的log.7成新的log.7，原来的log.7就自然失效。下面是一个失效处理的shell脚本，以供参考：

```
#!/bin/sh
# shell script ---  rotate_log.sh

if [ $# -ne 1 ]; then
   echo "Usage: $0 logname" 1&gt;&2
   exit 1
if

logfile=$1

mv $logfile.6 $logfile.7
mv $logfile.5 $logfile.6
mv $logfile.4 $logfile.5
mv $logfile.3 $logfile.4
mv $logfile.2 $logfile.3
mv $logfile.1 $logfile.2
mv $logfile $logfile.1
mysqladmin -u flush -pflushpass flush-logs     #执行mysqladmin flush-logs会打开一个日志
```

  该脚本以日志文件名为参数，执行方法如下：

```
% rotate_log.sh /usr/local/mysql/data/log
```

  注意，脚本中的mysqladmin命令是带有-u和-p参数的，因为我们进行日志刷新时需连接服务器。为确保安全，我们建立一个flush用户，密码为flushpass。该用户只有日志刷新的权限(reload权限)。创建该用户的语句如下：

```
GRANT RELOAD ON *.* TO 'flush'@'localhost' IDENTIFIED BY 'fulshpass';
```

设置好后，我们就可利用系统的自动处理机制定期运行该脚本以生成轮转日志。在Linux系统上的MySQL发行版中带有一个用来安装mysql-log-rotate日志轮转脚本的logrotate工具，如用RPM安装,则在/usr/share/mysql目录，如用二进制方式安装,则在MySQL安装目录的support-files目录，如用源码安装，则在安装目录的share/mysql目录中。

在windows平台下，不能在线更名，需停掉服务器，再进行。下面是一个进行日志更名的批处理文件：

```
@echo off
REM script name : rotate_log.bat

if not "%1" == "" goto ROTATE

   @echo Usage: rotate_log logname
   goto DONE

:ROTATE
set logfile=%1
erase %logfile%.7
rename %logfile%.6 %logfile%.7
rename %logfile%.5 %logfile%.6
rename %logfile%.4 %logfile%.5
rename %logfile%.3 %logfile%.4
rename %logfile%.2 %logfile%.3
rename %logfile%.1 %logfile%.2
rename %logfile% %logfile%.1
:DONE
```

该脚本的执行方法如下：

```
c:\rotate_log c:\mysql\data\log
```

- 以时间为依据对日志进行失效处理。该方法将定期删除超过给定时间的日志文件，适用于变更日志和二进制日志等文件名用数字编号标识的日志文件。下面是一个用Perl写成的处理脚本：

```
#!/usr/bin/perl -w

# script name: expire_log.pl
# Usage: expire_log.pl logfile ...

use strict
die "Usage: $0 logfile ...\n" if @ARGV == 0;
my $max_allowed_age = 7;      #max allowed age in days
foreach my $file (@ARGV)      #chack each argument
{
   unlink ($file) if -e $file && -M $file &gt;= $max_allowed_age;
}
exit(0);
```

该脚本需提供一个将被轮转的日志文件名作为参数，如：

```
% expire_log.pl /usr/local/mysql/data/log.[0-9]*
```

> 在给脚本参数时请小心，如给出*为参数，则会删除目录中所有更新时间大于7天的文件。

- 镜像机制。把日志文件镜像到所有的从服务器上。要使用镜像机制，你必须知道主服务器有多少个从服务器，哪些正在运行，并需依次连接每一个从服务器并发出show slave status语句以确定它正处理主服务器的哪个二进制日志文件(语句输出列表的Master_Log_File项)，只有所有的从服务器都不会用到的日志文件才能删除。删除方法是在主服务器上发出以下语句：

  ```
  mysql> PURGE MASTER LOGS TO 'last_log.xx';
  ```

  上面语句中的last_log.xx是所有从服务器已处理的最小编号日志文件。

4.6. MySQL服务器的一些优化配置

- 服务器的监听端口设置
  - TCP/IP端口3306是MySQL服务器默认的网络监听端口，如用--skip-networking选项启动服务器，则不监听TCP/IP端口。可用--port端口另行指定一个监听端口。如服务器主机有多个IP，还可用--bind-address选项对服务器在监听客户连接时使用的IP地址进行设定。
  - 在UNIX系统上，MySQL可在一个UNIX域套接字文件上监听有无本地客户在试图以localhost为主机名进行连接。默认的套接字文件是/tmp/mysql.sock，可用--socket选项指定另外一个套接字文件。
  - 在基于NT的Windows平台上，有-nt的MySQL服务器都支持命名管道。默认的命名管道是MySql，可用--socket选项另行指定。
- 启用或禁用LOAD DATA语句的LOCAL能力
  - 可在MySQL服务器编译时，用configure脚本的--enable-local-infile或--disable-local-infile选项把LOAD DATA语句的LOCAL能力设置为启用或禁用；
  - 在MySQL服务器启动是，可以用--local-infile或--disable-local-infile选项来启用或禁用服务器的LOCAL能力(在MySQL 4.0.2之前的版本里，要用--local-infile=0来禁用它)。

  如果在服务器端禁用了LOCAL的能力，则客户端就不能使用该功能；如服务器启用了LOCAL的能力，客户端默认也是禁止使用的，但可用mysql程序的--local-infile选项启用它。

- 国际化和本地化，国际化是指软件能够在世界多个国家地区使用，而本地化则是指可从国际化软件中选择一套适合本地区的语言和习惯的设置来使用。在MySQL中的国际化和本地化设置有以下几方面内容：

    - 时区，如果时区设置不对，则服务器显示的时间将会和当地时间有冲突。设置方法可通过mysqld_safe脚本的--timezone选项来设置，但最好还是在选项文件里设置，如:

    ```
    [mysqld_safe]
    timezone=US/Central
    ```

    - 配置显示信息的语言，MySQL能用多种语言来显示诊断信息与出错信息，默认是英语。查看share/mysql目录下有几个以语言名称作为目录名的目录就可知道有哪些语言可供选择。可用--language启动选项来指定语言，如--language=/usr/local/mysql/share/mysql/french。

    - 配置服务器的字符集，MySQL支持多种字符集，可在share/mysql/charsets目录下查询支持的字符集，也可用show variables like 'character_sets'来显示支持的字符集清单。MySQL把latin1作为默认的字符集。可在编译时用--with-charset指定另外一个字符集为默认字符集。如要增加另外的字符集支持，可用--with-extra-charasets选项进行添加。如：

    ```
    % ./configure --with-extra-charsets=latin1,gb2312,big5
    ```

    --with-extra-charsets有两个特殊的选项，一个是all，代表所有可用字符集；一个是complex，代表所有的复杂字符集(包括多字节字符集和有特殊排序规则的字符集)。

    服务器启动时，使用默认字符集，如需指定另外的字符集，需用--default-character-set选项指明。

    在MySQL 4.1以前，如果在创建好数据表后改变服务器的默认字符集，就需对索引重新排序才能保证索引键值能够正确反映出数据表记录在新字符集下的排列顺序。重新排序的操作命令如下：

    ```
    % myisamchk --recover --quick --set-character-set=gb2312       #在执行该语句需关闭服
    也可用：
    % mysqlcheck --repair --quick                                 #不需关闭服务器，适用
    或者用：
    mysql> REPLACE TABLE ... QUICK；
    ```

    在客户端，可用--default-character-set选项指定客户程序使用的字符集。--character-sets-dir选项可指出字符集文件的安装目录。

- 升级数据表到4.1，支持多字符集数据表。步骤如下：

1. 用mysqldump程序备份数据库：

```
% mysqldump -p -u root --all-databases --opt &gt; dumpfile.sql
--all-databases选项的作用是转储所有数据库；
--opt选项的作用是对转储文件进行优化。
```

2. 关闭服务器，升级MySQL服务器软件到4.1版。
3. 用备份文件重新加载数据表：

```
% mysql -p -u root &lt; dumpfile.sql
```

这样，字符集信息就被分配到每一个数据列中，此后，即使服务器改变了默认的字符集，各数据列的字符集也不会改变。当以后修改某个数据列的字符集时，服务器会自动重索引，以反映最新变化。

- 配置InnoDB表空间。InnoDB表空间在逻辑上是一个连接的存储区域，但实际上是由一个或多个磁盘文件组成。这些文件可以是普通的文件，也可以是一个未格式化的原始硬盘分区。InnoDB表空间通过一系列的配置选项来设置，其中最重要的有以下两个：

  为确保服务器每次启动时都能调用同样的选项，InnoDB的选项最好存放到选文件中。下面是一个例子：

  ```
  innodb_data_home_dir =
  innodb_data_file_path=/usr/loca/mysql/data/idbdata1:10M:autoextend:max:100M
  说明：
  InnoDB表空间文件默认存放到了MySQL的数据目录中，名字叫idbdata1；
  文件长度为10M；
  可自动扩展，以8M为步长扩展，如有多个数据文件，只允许最后一个文件可自动扩展；
  规定了最大的可扩展尺寸为100M。
  ```

  - innodb_data_home_dir，设置InnoDB表空间各组成文件的父目录，如果没有指出，则默认是MySQL的数据目录。
  - innodb_data_file_path，描述InnoDB主目录中各有关文件，包括文件名，文件长度和一些选项。各文件以分号分隔，各组成文件长度至少为10M。

  把选项写入选项文件后，启动服务器就可自动创建和初始化InnoDB表空间。

  利用原始磁盘分区作为InnoDB表空间可创建一个非常大的表空间，不受操作系统单文件最大容量的限制。并且能有效减少磁盘碎片的产生。要使用原始磁盘分区，需作如下配置：

  - 首先，要进行初始化，在选项文件的[mysqld]中配置：

    ```
    innodb_data_home_dir=
    innodb_data_file_path=/dev/hda1:10Gnewraw     #初始化/dev/hda1这个10G容量的分区
    ```

启动服务器，服务器会对这个10G的分区进行初始化。

- 接着，关闭服务器，修改配置文件，把newraw改为raw，如：

```
innodb_data_home_dir=
innodb_data_file_path=/dev/hda1:10Graw
```

重新启动服务器，MySQL就会以读/写方式使用该表空间了。在windows平台上配置InnoDB表空间时，windows路径名中的反斜杠可以写成单个的斜线字符(/)。也可写成两个反斜杠(\)。如：

```
innodb_data_home_dir=
innodb_data_file_path=c:/mysql/data/ibdata1:10M;d:/ibdata2:20M
```

默认情况下，InnoDB的日志文件会存储在MySQL的数据目录，文件名以ib开头。一旦完成InnoDB表空间的初始化，就不能改变组成文件的大小，但可通过添加数据文件或设置自动扩展来增加表空间容量。如需通过增加文件的方法扩大表空间的容量，可按以下步骤进行：

1. 关闭正在运行的MySQL服务器
2. 如果InnoDB表空间的最后一个组成文件是可自扩展的，就要先把它改变成一个固定长度文件才能把另一个文件添加到它后面。方法是先计算出该文件的近似大小，重新设置，如：

   ```
   innodb_data_file_path=ibdata1:100M:autoextend
   改成：
   innodb_data_file_path=ibdata1:150M
   ```

3. 把新的组成文件添加到文件清单的末尾，该文件可以是普通文件，也可以是一个原始硬盘分区。
4. 重启服务器。

还有一种方法重新配置InnoDB表空间，就是先备份，再重新配置，最后重新加载备份。具体步骤如下：

1. 使用mysqldump备份整个InnoDB数据库；
2. 关闭服务器，删除所有InnoDB表空间、InnoDB日志文件 及InnoDB数据表的.frm文件；
3. 重新配置InnoDB表空间；
4. 配置完成后，用备份文件重载数据，生成新的InnoDB数据表。

4.7. 优化服务器

MySQL服务器为我们提供了丰富的参数，以调整服务器满足不同环境的要求。下面分别讨论一下这些参数：

- 服务器参数变量的设置。MySQL服务器参数可在服务器启动时设置，在MySQL4.0.3及以后的版本中，有些参数也允许在线设置。在MySQL4.0.2及以后的版本里，可以把一个变量名视为一个选项名来设置。如数据表缓冲区的尺寸由服务器参数talbe_cache来设置。如果需把它设置为128,则可以在命令行里增加

```
--table_cache=128
```

  也可在选项文件中设置：

```
[mysqld]
table_cache=128
```

  在命令行选项中'_'可写'-'，变成：

```
--table-cache=128      #这种写法更像一个标准选项
```

  还有一种是使用--set-variable或-O选项，如：

```
--set-variable=table_cache=128
or
-O table_cache=128
在选项文件中可写成：
[mysqld]
set-variable=table_cache=128
```

  服务器参数分为全局级和会话级两个级别。全局级参数将影响整个服务器，会话级参数则只影响某给定客户连接上的工作。如果某个变量同时存在于两个级别，则服务器在客户建立连接时用全局变量的值去初始化相应的会话级参数，一旦客户连接建立起来后，对全局参数所作的修改不会影响到相应的会话级参数当前值。设置全局参数和会话级参数的语句：

```
全局级：
mysql> SET GLOBAL variable = value;
mysql> SET @@GLOBAL.variable = value;
会话级：
mysql> SET SESSION variable = value;
mysql> SET @@SESSION.variable = value;
默认不带级别限定符的SET语句修改的参数属会话级，如：
mysql> SET variable = value;
mysql> SET @@variable = value;
可用一条SET语句设置多个参数，参数间用逗号分隔，如：
SET SESSION variable = value1,value2,value3;
```

SESSION和LOCAL是同义语，可用LOCAL代替SESSION，如：@@LOCAL

具备SUPER权限才能设置全局参数，新设置值的效力将持续到该参数被再次修改或服务器退出。设置会话级参数不用特殊的权限，新设置值的效力将持续到该值被再次修改或连接断开。显示参数的语句如下：

```
SHOW GLOBAL VARIABLES;
SHOW GLOBAL VARIABLES LIKE 'TEST';
SHOW SESSION VARIABLES;
SHOW SESSION VARIABLES LIKE 'TEST';
如不带限定符，则返回会话级参数，如会话级参数不存在则返回全局级参数。
也可用命令行方式显示服务器参数变量，如：
% mysqladmin variables
```

- 下面介绍一些MySQL服务器通用的参数变量：
    - back_log，当多个客户同时连接服务器时，客户处理过程需进入一个队列排队等待服务器处理。该值定义服务器等待处理队列长度的最大值，如果站点访问量大，需加大该值。
    - delayed_queue_size，在实际插入数据表前，来自insert delayed语句的数据行会进入一个队列等待服务器处理。该值定义该队列能容纳的数据行的最大个数。当队列满时，会阻塞后续的语句。加大该值能提高insert delayed语句的执行速度。
    - flush_time，自动存盘间隔。如果系统经常死机或重启，把这个变量设置为一个适当的非零值，使MySQL服务器每隔flush_time称去刷新一次数据表缓冲区，将其中的信息写入磁盘。这将导致系统性能下降，但可减少数据表被破坏或丢失数据的概率。在命令行上用--flush选项启动服务器可使数据表在每次修改后被自动存盘。
    - key_buffer_size，用来容纳索引块的缓冲区的长度。加大该值可加快索引创建和修改操作的速度，该索引缓冲区越大，在内存中找到键值的可能性就越大，读盘次数就越少。MySQL3.23前的版本里，该参数叫key_buffer。3.23版本之后，两种叫法都可以。
    - max_allowed_packet，服务器与客户程序之间通信时使用的缓冲区在最大值。MySQL 4版本之前，该最大值可取16MB，MySQL 4版本以后，该值的最大值是1GB。如果客户端与服务器需传送大容量的数据，如BLOB或TEXT值，就要加大该值。客户端也有一个同名的变量，默认是16MB，该值也要加大。客户端的启动命令为：

      ```
      % mysql --set-variable=max_allowed_packet=64M
      ```

    - max_connections，允许同时打开的连接数，如果站点繁忙，需加大该值。
    - table_cache，数据表缓存区的尺寸。加大该值可使服务器能够同时打开更多的数据表，从而减少文件打开/关闭操作的次数。

注意：加大max_connections和table_cache参数的值，会使服务器占用更多的文件描述符。运行多个服务器可绕过该限制。对一些分配给每个客户的资源变量，设置时不能过大，因为当连接数快速增长时会很快耗尽服务器的资源，造成服务器性能下降。

- InnoDB处理程序变量：
  - innodb_buffer_pool_size，InnoDB数据库缓冲池的大小，如果有足够的内存，可把该值设置得大些以减少读盘操作。
  - innodb_log_file_size和innodb_log_files_in_group，前者设置日志文件的长度，后者设置日志文件的个数。InnoDB日志文件的总长度是两者的乘积，它的总长度不得超过4GB。

4.8. 运行多个MySQL服务器

需运行多个服务器的原因有很多，比如上面提到的可绕过最大文件描述符的限制，还有是进行版本测试和提供专用服务等。运行多个服务器比运行单个服务器复杂很多，需注意以下问题：

- 在安装不同版本的程序时，需分开目录存放程序和数据目录。如果同一版本的服务器软件，则程序目录可一样，但数据目录则要不同。可用--basedir=dir_name和--datadir=dir_name两个启动选项指时这两个目录。
- 要为不同的服务器指定不时的--port=port_name(TCP/IP监听端口)，--socket=file_name(套接字文件名)和--pid-file=file_name(进程ID文件)值。
- 如果激活了日志功能，就要为不同的服务器指定不同的日志文件名。
- 在Windows平台上，被安装为服务的多个MySQL服务器必须有不同的服务名。

多服务器环境下选项文件的配置方法：

- 使用--defaults-file选项指定每个选项文件，这样，每个服务器就不会去读/etc/my.cnf这些配置文件，而会使用你所指定的配置文件。
- 可把一些公共的选项放到/etc/my.cnf文件里，再用--defaults-extra-file选项指出特定服务器的特定选项文件。这样就不用在所有的配置文件时重复公共的选项。
- 用mysql_multi脚本启动服务器，它允许我们把所有的选项放到同一个选项文件里。每一个服务器对应该文件中的一个选项组。

  下面介绍用mysql_multi脚本启动多服务器的方法。

  1. 为每个服务器编一个编号xxx，对应选项文件的[mysqldxxx]选项组。mysql_multi本身要用到的选项可放到[mysqld_multi]里。这样/etc/my.cnf选项配置文件看起来就象下面这样：

```
[mysqld001]
basedir=/usr/local/mysql/001
datadir=/usr/local/mysql/001/data
mysqld=/usr/local/mysql/001/bin/mysqld_safe
socket=/usr/local/mysql/001/mysql.sock
port=3306
local-infile=1
user=mysqladm
log=log
log-update=update-log
innodb_data_file_path=ibdata1:10M

[mysqld002]
basedir=/usr/local/mysql/002
datadir=/usr/local/mysql/002/data
mysqld=/usr/local/mysql/002/bin/mysqld_safe
socket=/usr/local/mysql/002/mysql.sock
port=3307
local-infile=1
user=mysqladm
log=log
log-update=update-log
innodb_data_file_path=ibdata1:10M

...
```

2. 配置好选项文件后，就可用以下命令启动服务器：

```
% mysqld_multi --no-log start 001,002
#启动001和002两个服务器，并把启动信息发送到控制台，也可用区间的形式给出服务器编号
```

   用以下命令可查看服务器状态：

```
% mysqld_multi --no-log --user=root --password=password report 001
```

   可用以下命令停止MySQL服务器：

```
% mysqld_multi --no-log --user=root --password=password stop 001
```

- 在windows平台下运行多个MySQL服务器的方式有两种，一种是运行同一个MySQL程序的两个实例，一种是运行多个windows服务，下面分别介绍：
    - 第一种情况需设置两个选项文件，指定不同的数据目录，如：

```
c:\mysql\my.cnf1

[mysqld]
basedir=c:/mysql
datedir=c:/mysql/data1
port=3306

c:\mysql\my.cnf2

[mysqld]
basedir=c:/mysql
datadir=c:/mysql/data2
port=3307
```

在启动服务器时，用--defaults-file选项指出选项文件即可。如：

```
c:\&gt; mysqld --defaults-file=c:\mysql\my.cnf1
c:\&gt; mysqld --defaults-file=c:\mysql\my.cnf2
```

- 在MySQL 4.0.2版本开始，可以把MySQL安装为一个服务，并可指定一个服务名，如：

  ```
  c:\&gt; mysql-nt --install service_name
  ```

  在MySQL 4.0.3开始，安装服务还支持--defaults-file=file_name选项

  这样，我们就可把MySQL安装为一系列不同的服务，如果不指定service_name，则安装的服务名默认为MySql，如果指定service_name，则安装的服务名为指定的service_name，并对应选项文件中的[service_name]选项组。以默认服务名运行的服务器还支持一个名为MySql的命名管道，而明确给出服务名的服务器将只监听TCP/IP连接而不支持命名管道--除非还用socket选项明确指定一个套接字文件。

  移除服务需先用mysqladmin shutdown命令停掉服务器，再执行以下命令：

  ```
  c:\&gt; mysql-nt --remove                 #移除默认的服务
  c:\&gt; mysql-nt --remove service_name    #移除指定服务
  ```

4.9. MySQL服务器镜像配置

通过镜像机制可把数据从一个地方复制到另一个地方，并能实现同步两个或多个地方的数据。MySQL服务器也支持镜像，大提高数据的安全性和稳定性。下面介绍一下MySQL数据中的镜像机制：

- 在镜像关系中，一个MySQL服务器扮演主服务器角色，另外一个或多个服务器扮演从服务器角色，从服务器中的数据和主服务器中的数据完全一样。
- 在镜像建立之前，主服务器和从服务器必须进行一次完全同步。同步之后，在主服务器上所做的操作将会在从服务器上再实现，主服务器上的操作不是直接作用于从服务器上的。
- 负责在主、从服务器上传输各种修改动作的媒介是主服务器上的二进制变更日志，该日志记录着主服务器上所有的操作动作。因此，主服务器必须激活二进制日志功能。
- 从服务器必须有足够的权限从主服务器上接收二进制日志文件。镜像协调信息记录从服务器的进展情况，包括，从服务器正在读取的二进制变更日志文件名和它在该文件里的当前读写位置。

- 每个主服务器可以有多个从服务器，但每个从服务器只能有一个主服务器。但MySQL服务器允许把一个从服务器作为另一个从服务器的主服务器，这样就可创建一个镜像服务器链。

镜像机制在MySQL中还是一个新生事物，最早实现于3.23.15版。各版本间的镜像能力有差异，一般来说，建议大家尽量使用最新的版本，下面列举了不同版本的MySQL服务器在镜像机制方面的兼容规则：

- 3.23.x系统版本的从服务器不能与4.x系统版本的主服务器通信。
- 4.0.0版本的从服务器只能与4.0.0版本的主服务器通信。
- 4.0.1或更高版本的从服务器既能与3.23.x系统版本的主服务器通信，也能与4.x系统版本的主服务器通信。但后一种情况要求主服务器的版本号等于或大于从服务器的版本号。

一般来说，建议遵循以下原则：

- 要尽可能地让主服务器和从服务器都使用同一版本系统。
- 在选定系统后，尽量使用该系统的最新版本。

建立主从镜像服务器的步骤：

- 确定主从服务器的镜像ID号，主从服务器的ID号不能相同。在启动主从服务器时，用--server_id启动选项给出其ID。
- 从服务器必须在主服务器上有一个具备足够的权限的帐户，从服务器将使用该帐户去连接主服务器并请求主服务器把二进制变更日志发送给它。可用以下命令创建这个帐户：

```
mysql> GRANT REPLICATION SLAVE ON *.* TO 'slave_user'@'slave_host' IDENTIFIED BY 'sla
#REPLICATION权限只MySQL4.0.2后版本，之前的版本请用FILE权限。
```

- 把主服务器上的数据库文件拷贝到从服务器上完成最初同步工作。也可用备份后再加载的方法。在MySQL 4.0.0及以后版本里，还可用在主服务器上运行LOAD DATA FROM MASTER语句来建立从服务器。但有约束条件：
    - 数据表要全部是MyISAM表
    - 为发出这条指令而在连接从服务器时使用的帐户必须有SUPER权限。
    - 从服务器用来连接主服务器的帐户必须具备RELOAD和SUPER权限。注意，这是一个主服务器上的帐户，而用来发出LOAD DATA FROM MASTER语句的帐户是一个从服务器上的帐户。
    - LOAD DATA FROM MASTER语句在执行时需申请一个全局性的读操作锁，这个锁在语句执行期间阻塞主服务器上一切的写操作。

无论用哪种方法同步数据，都要确保在开始制作备份到给主服务器重新配置好二进制日志功能这段时间，不能在主服务器上发生修改操作。

- 关闭服务器。
- 对主服务器的配置进行修改--把它的镜像ID告诉它并激活其二进制日志功能。在主服务器要读取的选项文件内增加以下内容：

```
[mysqld]
server-id=master_server_id
log-bin=binlog_name
```

- 重新启动主服务器，从现在开始，它将把客户对数据库的修改操作记录到二进制变更日志里。如果在此之前已经激活了二进制日志功能，则要在重启前把二进制变更日志备份下来，在重启后再发一条RESET MASTER语句去清除现有的二进制变更日志。
- 关闭从服务器。
- 对从服务器进行配置，使它知道自已的镜像ID，到哪里去找主服务器以及如何去连接主服务器。配置内容如下：

```
[mysqld]
server-id=slave_server_id
master-host=master_host
master-user=slave_user          #在主服务器上为从服务器建立的帐户
msater-password=slave_pass      #在主服务器上为从服务器建立的帐户的密码
master-connet-retry=30         #设置连接重试间隔，默认为60秒
master-retry-count=100000       #设置重试次数，默认为86400次
注：最后两个选项在网络连接不可靠时设置
```

- 重新启动从服务器。从服务器用两个信息源来确定它自已在镜像工作中的进度位置：一个是数据目录中的master.info文件，另一个是启动选项所给定的配置信息。第一次启动从服务器时，master.info文件不存在，从服务器会根据选项文件中给出的各种master-xxx选项值去连接主服务器。一旦连接成功，从服务器会创建一个master.info文件以保存各种连接参数和它自已的镜像工作状态。如果以后再重启从服务器，从服务器会优先读取该文件，而不是选项文件。所以如果你修改了选项文件的内容，想该选项生效就要删除master.info文件并重启从服务器。

以上步骤是镜像所有数据库的操作过程，如果我们想把mysql权限数据保留在主服务器上，排除在镜像机制外的话，可用在选项文件的[mysqld]中加入--binlog-ignore-db=mysql选项，这样，mysql数据库上的操作就不会记录在二进制变更日志里。如要排除其它数据库，只要增加多几个该选项即可。

通过以下几个命令可监控和管理主从服务器：

- SLAVE STOP，SLAVE START用于挂起来恢复从服务器上镜像，如当备份时，可用该语句让从服务器暂时停止镜像活动。

- SHOW SLAVE STATUS，在从服务器上查看其镜像协调信息，这些信息可以用来判断哪些二进制变更日志已经不再使用。
- PURGE MASTER，在主服务器上对二进制变更日志进行失效处理。删除所有从服务器都不再使用的二进制变更日志。
- CHANGE MASTER，在从服务器上修改镜像参数。如正在读取主服务器上哪个二进制变更日志，正在写哪个中继日志文件等。

在MySQL4.0.2之后版本中，镜像机制中的从服务器由两个内部线程组成：

- 一个叫“I/O线程”，负责与主服务器通信，请求主服务器发送二进制变更日志，并把接收到的数据修改命令写入某个中继日志文件；用SLAVE STOP IO_THREAD或SLAVE START IO_THREAD可挂起或恢复该线程。
- 另一个叫“SQL线程”，负责从中继日志中读取数据修改命令并执行。同理，用SLAVE STOP SQL_THREAD或SLAVE START SQL_THREAD可挂起或恢复该线程。

中继日志文件默认的文件为hostname-relay-bin.nnn和hostname-relay-bin.index。可用从服务器的--relay-log和--relay-log-index选项修改。在从服务器中还有一个relay-log.info中继信息文件，可用--relay-log-info-file启动选项修改文件名。

Chapter 5. 数据库安全

安全是一个过程，而不是一个方法，它贯穿在我们使用和维护MySQL数据库的过程中。这不单是系统管理员工作，用户也要有安全的意识，使安全问题得到有效控制。MySQL服务器的安全问题可分为内部安全和外部安全两部份。内部安全问题大都与系统文件有关，我们需确保MySQL程序文件和数据文件的安全。外部安全是指通过网络连接到服务器的安全问题，应该只允许合法用户访问数据库，在一些情况下还可用SSL加密信息传输通道。下分别介绍内部安全和外部安全的防范措施。

5.1. 保护MySQL安装程序文件

- 在重设置文件权限时，请先关闭数据库服务器。
- 用以下命令把MySQL安装程序目录的属主和所属组设置为MySQL管理员帐号的用户名和用户组名。

  ```
  % chown -R mysql.mysql /usr/local/mysql
  ```

  另外一种方法是把除数据目录外的所有目录属主设置为root所有，如：

```
% chown -R root.mysql /usr/local/mysql
% chown -R mysql.mysql /usr/local/mysql/data
```

- 设置安装目录及各有关子目录的权限，允许管理员进行所有操作，只允许其他人进行读和执行访问，设置命令如下：

```
#设置mysql目录
% chmod 755 /usr/local/mysql
or
% chmod u=rwx,go=rx /usr/local/mysql
#设置mysql/bin目录
% chmod 755 /usr/local/mysql/bin
or
% chmod u=rwx,go=rx /usr/local/mysql/bin
#设置mysql/libexec目录
% chmod 700 /usr/local/mysql/libexec
or
% chmod u=rwx,go-rwx /usr/local/mysql/libexec
```

- 把数据目录及目录中的所有子目录和文件设置为只允许MySQL管理员访问。

```
% chmod -R go-rwx /usr/local/mysql/data
```

  如果数据目录下有选项文件或套接字文件，并一些客户需访问这些文件，则可用以下的权限设置，使客户在没有读权限的前提下使用这些文件：

```
% chmod go+x /usr/local/mysql/data
```

- mysql.sock套接字文件一般放以/tmp目录下，要确保该目录设置了粘着位，使自户只能删除自已创建的文件，不能删除其他用户创建的文件。/etc/my.cnf中公共选项文件，是对所有用户可读的，所以不应把一些敏感信息保存在里面。.my.cnf是用户专用选项文件，要确保只有该用户有权访问。
- 这样设置以后，只有MySQL管理员才能启动服务器。

5.2. 权限表

MySQL服务器通过权限表来控制用户对数据库的访问，权限表存放在mysql数据库里，由mysql_install_db脚本初始化。这些权限表分别user，db，table_priv，columns_priv和host。下面分别介绍一下这些表的结构和内容：

- user权限表：记录允许连接到服务器的用户帐号信息，里面的权限是全局级的。
- db权限表：记录各个帐号在各个数据库上的操作权限。
- table_priv权限表：记录数据表级的操作权限。

- columns_priv权限表：记录数据列级的操作权限。
- host权限表：配合db权限表对给定主机上数据库级操作权限作更细致的控制。这个权限表不受GRANT和REVOKE语句的影响。

大家注意到，以上权限没有限制到数据行级的设置。在MySQL只要实现数据行级控制就要通过编写程序(使用GET-LOCK()函数)来实现。

MySQL的版本很多，所以权限表的结构在不同版本间会有不同。如果出现这种情况，可用mysql_fix_privilege_tables脚本来修正。运行方式如下：

```
% mysql_fix_privilege_tables rootpassword            #这里要给出MySQL的root用户密码
```

最好一下子升级到MySQL 4.0.4版本，因为4.0.2和4.0.3的db表没有Create_tmp_table_priv和Lock_tables_priv权限。

MySQL的权限表定义了两部份内容，一个部份定义权限的范围，即谁(帐户)可以从哪里(客户端主机)访问什么(数据库、数据表、数据列)；另一部份定义权限，即控制用户可以进行的操作。下面是一些常用的权限介绍，可直接在GRANT语句中使用。

- CREATE TEMPORARY TABLES，允许创建临时表的权限。
- EXECUTE，允许执行存储过程的权限，存储过程在MySQL的当前版本中还没实现。
- FILE，允许你通过MySQL服务器去读写服务器主机上的文件。但有一定限制，只能访问对任何用户可读的文件，通过服务器写的文件必须是尚未存在的，以防止服务器写的文件覆盖重要的系统文件。尽管有这些限制，但为了安全，尽量不要把该权限授予普通用户。并且不要以root用户来运行MySQL服务器，因为root用户可在系统任何地方创建文件。
- GRANT OPTION，允许把你自已所拥有的权限再转授给其他用户。
- LOCK TABLES，可以使用LOCK TABLES语句来锁定数据表
- PROCESS，允许你查看和终止任何客户线程。SHOW PROCESSLIST语句或mysqladmin processlist命令可查看线程，KILL语句或mysqladmin kill命令可终止线程。在4.0.2版及以后的版本中，PROCESS权限只剩下查看线程的能力，终止线程的能力由SUPER权限控制。
- RELOAD，允许你进行一些数据库管理操作，如FLUSH，RESET等。它还允许你执行mysqladmin命令：reload，refresh，flush-hosts，flush-logs，flush-privileges，flush-status，flush-tables和flush-threads。
- REPLICATION CLIENT，允许查询镜像机制中主服务器和从服务器的位置。

- REPLICATION SLAVE，允许某个客户连接到镜像机制中的主服务器并请求发送二进制变更日志。该权限应授予从服务器用来连接主服务器的帐号。在4.0.2版这前，从服务器是用FILE权限来连接的。

- SHOW DATABASES，控制用户执行SHOW DATABASES语句的权限。

- SUPER，允许终止线程，使用mysqladmin debug命令，使用CHANGE MASTER，PURGE MASTER LOGS以及修改全局级变量的SET语句。SUPER还允许你根据存放在DES密钥文件里的密钥进行DES解密的工作。

user权限表中有一个ssl_type数据列，用来说明连接是否使用加密连接以及使用哪种类型的连接，它是一个ENUM类型的数据列，可能的取值有：

- NONE，默认值，表示不需加密连接。

- ANY，表示需要加密连接，可以是任何一种加密连接。由GRANT的REQUIRE SSL子句设置。

- X509，表示需要加密连接，并要求客户提供一份有效的X509证书。由GRANT的REQUIRE X509子句设置。

- SPECIFIED，表示加密连接需满足一定要求，由REQUIRE子句的ISSUER，SUBJECT或CIPHER的值进行设置。只要ssl_type列的值为SPECIFIED，则MySQL会去检查ssl_cipher(加密算法)、x509_issuer(证书签发者)和x509_subject(证书主题)列的值。这几列的列类型是BLOB类型的。

user权限表里还有几列是设置帐户资源使用情况的，如果以下数据列中的数全为零，则表示没有限制：

- max_connections，每小时可连接服务器的次数。

- max_questions，每小时可发出查询命令数。

- max_updates，每小时可以发出的数据修改类查询命令数。

设置权限表应注意的事项：

- 删除所有匿名用户。

- 查出所有没有口令用户，重新设置口令。可用以下命令查询空口令用户：

  ```
  mysql> SELECT host,user FROM user WHERE password = '';
  ```

- 尽量不要在host中使用通配符。

- 最好不要用user权限表进行授权，因为该表的权限都是全局级的。

- 不要把mysql数据库的权限授予他人，因为该数据库包含权限表。

- 使用GRANT OPTION权限时不要滥用。
- FILE权限可访问文件系统中的文件，所以授权时也要注意。一个具有FILE权限的用户执行以下语句就可查看服务器上全体可读的文件：

```
mysql> CREATE TABLE etc_passwd(pwd_entry TEXT);
mysql> LOAD DATA INFILE '/etc/passwd' INTO TABLE etc_passwd;
mysql> SELECT * FROM etc_passwd；
```

  如果MySQL服务器数据目录上的访问权限设置得不好，就会留下让具有FILE权限的用户进入别人数据库的安全漏洞。所以建议把数据目录设置成只能由MySQL服务器读取。下面演示一个利用具有FILE权限的用户读取数据目录中文件权限设置不严密的数据库数据的过程：

```
mysql> use test;
mysql> create table temp(b longblob);
mysql> show databases                   #显示数据库名清单，--skip-show-database可禁止该功
mysql> load data infile './db/xxx.frm' into table temp fields escaped by '' lines ter
mysql> select * from temp into outfile 'xxx.frm' fields escaped by '' lines terminate
mysql> delete from temp;
mysql> load data infile './db/xxx.MYD' into table temp fields escaped by '' lines ter
mysql> select * from temp into outfile 'xxx.MYD' fields escaped by '' lines terminate
mysql> delete from temp;
mysql> load data infile './db/xxx.MYI' into table temp fields escaped by '' lines ter
mysql> select * from temp into outfile 'xxx.MYI' fields escaped by '' lines terminate
mysql> delete from temp;
```

  这样，你的数据库就给人拷贝到本地了。如果服务器是运行在root用户下，那危害就更大了，因为root可在服务器上做任何的操作。所以尽量不要用root用户来运行服务器。
- 只把PROCESS权限授予可信用户，该用户可查询其他用户的线程信息。
- 不要把RELOAD权限授予无关用户，因为该权限可发出FLUSH或RESET语句，这些是数据库管理工具，如果用户不当使用会使数据库管理出现问题。
- ALTER权限也不要授予一般用户，因为该权限可更改数据表。

GRANT语句对权限表的修改过程：

- 当你发送一条GRANT语句时，服务器会在user权限表里创建一个记录项并把你用户名、主机名和口令记录在User、Host和Password列中。如果设置了全局权限，由把该设置记录在相在的权限列中。
- 如果在GRANT里设置了数据库级权限，你给出的用户名和主机名就会记录到db权限表的User和Host列中，数据库名记录在Db列中，权限记录到相关的权限列中。
- 接着是到数据表和数据列级的权限设置，设置方法和上面的一样。服务器会把用户名、主机名、数据库名以及相应的数据表名和数据列名记录到数据表中。

删除用户权限其实就是把这些权限表中相应的帐号记录全部删除即可。

5.3. 建立加密连接

加密连接可提高数据的安全性，但会降低性能。要进行加密连接，必须满足以下要求：

- user权限表里要有相关的SSL数据列。如果安装的MySQL服务器是4.0.0版的，user权限表已包含相关的SSL数据列，否则，我们也可用mysql_fix_privilege_tables脚本升级权限表。

- 服务器和客户程序都已经编译有OpenSSL支持。首先要安装openssl，在编译时MySQL服务器时加--with-vio和--with-openssl选项加上openssl支持。可用以下语句查询服务器是否支持SSL：

  ```
  mysql> show variables like 'have_openssl';
  ```

- 在启动服务器时用有关选项指明证书文件和密钥文件的位置。在建立加密连接前，要准备三个文件，一个CA证书，是由可信赖第三方出具的证书，用来验证客户端和服务器端提供的证书。CA证书可向商业机构购买，也可自行生成。第二个文件是证书文件，用于在连接时向对方证明自已身份的文件。第三个文件是密钥文件，用来对在加密连接上传输数据的加密和解密。MySQL服务器端的证书文件和密钥文件必须首先安装，在sampdb发行版本的ssl目录里有几个供参考的样本文件：ca-cert.pem(CA证书)，server-cert.pem(服务器证书)，server-key.pem(服务器公共密钥)。把这几个文件拷贝到服务器的数据目录中，再在选项文件里加上以下内容：

  ```
  [mysqld]
  ssl-ca=/usr/local/mysql/data/ca-cert.pem
  ssl-cert=/usr/local/mysql/data/server-cert.pem
  ssl-key=/usr/local/mysql/data/server-key.pem
  ```

  重启服务器，使配置生效。

- 要想让某个客户程序建立加密连接，必须在调用这个客户程序时用有关选项告诉它在哪里能找到其证书文件和密钥文件。在sampdb发行版的ssl目录中提供了client-cert.pem(客户证书文件)，client-key.pem(客户密钥文件)，CA证书与服务器使用同样的ca-cert.pem。把他们拷贝到个人目录下，并在.my.cnf选项文件中指出文件位置，如：

  ```
  [mysql]
  ssl-ca=/home/mysql/ca-cert.pem
  ssl-cert=/home/mysql/client-cert.pem
  ssl-key=/home/mysql/client-key.pem
  ```

  配置完成后，调用mysql程序运行\s或SHOW STATUS LIKE 'SSL%'命令，如果看到SSL:的信息行就说明是加密连接了。如果把SSL相关的配置写进选项文件，则默认是加密连接的。也可用mysql程序的--skip-ssl选项取消加密连接。如果用命令行方式启用加密连接可以这样写：

```
% mysql --ssl-ca=ca-cert.pem --ssl-cert=client-cert.pem --ssl-key=client-key.pem
```

可用GRANT语句的REQUIRE SSL选项来强制用户使用加密连接。

使用sampdb发行版的证书可以建立一个加密连接，但由于该文件已公开，所以安全性不好，我们可以在测试成功后自行建立证书或购买商业证书，以提高安全性。如何自行建立SSL证书的文档在sampdb发行版的ssl/README文件里有说明。

Chapter 6. 数据库的备份、维护和修复

数据库在运行中，会因为人为因素或一些不可抗力因素造成数据损坏。所以为了保护数据的安全和最小停机时间，我们需制定详细的备份/恢复计划，并定期对计划的有效性进行测试。本章结合MySQL服务器的运行机制和所提供的工具，介绍如何进行数据库的备份、维护和修复。

以下是几点防范的措施：

- 制定一份数据库备份/恢复计划，并对计划进行仔细测试。
- 启动数据库服务器的二进制变更日志，该功能的系统开销很小(约为1%)，我们没有理由不这样做。
- 定期检查数据表，防范于未燃。
- 定期对备份文件进行备份，以防备份文件失效。
- 把MySQL的数据目录和备份文件分别放到两个不同的驱动器中，以平衡磁盘I/O和增加数据的安全。

6.1. 检查/修复数据表

对数据表进行维护最好通过发出CHECK TABLE(检查数据表)或REPAIRE TABLE(修复数据表)命令来做，这样MySQL服务器自动进行表锁定以协调数据表中数据的读写一致性问题。也可用myisamchk工具来做数据表的维护，但它直接访问有关的数据表文件，不通过服务器，所以需人为地协调数据表数据的读写一致性问题。使用myisamchk检查数据表的具体操作步骤如下：

- 以mysql客户端程序连接服务器，并发出LOCK TABLE命令，以只读方式锁住数据表。

```
% mysql
mysql> use db
mysql> LOCK TABLE table_name READ;      #以只读方式锁定表
mysql> FLUSH TABLE table_name;        #关闭数据表文件，并把内存中的信息写入磁盘
```

- 保持上面的状态不退出，另开一个shell窗口，用以下命令维护(检查)数据表。

```
% myisamchk table_name
```

> 如果不保持上面状态，退出mysql会话，则表锁定会自动取消。

- 维护完成，切换回mysql状态的shell窗口，发出以下命令解除表锁定。

```
mysql> UNLOCK TABLES;
```

使用myisamchk修复数据表的具体操作步骤如下：

- 进行修复操作需以读/写方式锁定数据表，命令如下：

```
% mysql
mysql> use db
mysql> LOCK TABLE table_name WRITE;     #以读/写方式锁定数据表
mysql> FLUSH TABLE table_name;
```

- 保持mysql客户端连接状态，切换到第二个shell窗口，运行修复命令：

```
% myisamchk --recover table_name
```

> 运行修复命令前最好先备份一下数据文件。

- 修复完成后，切换回mysql客户端连接窗口，运行以下命令解除数据表锁定：

```
mysql> FLUSH TABLE table_name;     #使服务器觉察新产生的索引文件
mysql> UNLOCK TABLE;
```

还可用以下命令锁定所有表，锁定后，所有用户就只能读不能写数据，这样就可使我们能安全地拷贝数据文件。

```
mysql> FLUSH TABLES WITH READ LOCK;
```

下面是解除锁语句：

```
mysql> UNLOCK TABLES;
```

6.2. 备份数据库

定期的备份可使我们数据库崩溃造成的损失大大降低。在MySQL中进行数据备份的方法有两种，一种是使用mysqldump程序，一种是使用mysqlhotcopy、cp、tar或cpio等打包程序直接拷贝数据库文件。mysqldump程序备份数据库较慢，但它生成的文本文件便于移植。使用mysqlhotcopy等程序备份速度快，因为它直接对系统文件进行操作，需人为协调数据库数据的备份前后一致性。

- 使用mysqldump备份数据库其实就是把数据库转储成一系列CREATE TABLE和INSERT语句，通过这些语句我们就可重新生成数据库。使用mysqldump的方法如下：

  ```
  % mysqldump --opt testdb | gzip &gt; /data/backup/testdb.bak
  #--opt选项会对转储过程进行优化，生成的备份文件会小一点，后的管道操作会进行数据压缩
  % mysqldump --opt testdb mytable1,mytable2 | gzip &gt; /data/backup/testdb_mytable.ba
  #可在数据库后接数据表名，只导出指定的数据表，多个数据表可用逗号分隔
  ```

  > --opt选项还可激活--add-drop-table选项，它将会在备份文件的每条CREATE TABLE前加上一条DROP TABLE IF EXISTS语句。这可方便进行数据表的更新，而不会发生“数据表已存在”的错误。

  用mysqldump命令还可直接把数据库转移到另外一台服务器上，不用生成备份文件。重复执行可定期更新远程数据库。

  ```
  % mysqladmin -h remote_host create testdb
  % mysqldump --opt testdb | mysql -h remote_host testdb
  另外还可通过ssh远程调用服务器上的程序，如：
  % ssh remote_host mysqladmin create testdb
  % mysqldump --opt testdb | ssh remote_host mysql testdb
  ```

- 通过直接拷贝系统文件的方式备份数据库，在备份时，要确保没有人对数据库进行修改操作。要做到这点，最好关闭服务器。如果不能关闭的，要以只读方试锁定有关数据表。下面是一些示例：

  ```
  % cp -r db /backup/db                 #备份db数据库到/backup/db目录
  % cp table_name.* /backup/db          #只备份table_name数据表
  % scp -r db remotehot:/usr/local/mysql/data      #用scp把数据库直接拷贝到远程服务器
  ```

  > 在把数据库直接拷贝到远程主机时，应注意两台机器必须有同样的硬件结构，或者将拷贝的数据表全部是可移植数据表类型。

- 使用mysqlhotcopy工具，它是一个Perl DBI脚本，可在不关闭服务器的情况下备份数据库，它主要的优点是：

  - 它直接拷贝文件，所以它比mysqldump快。
  - 可自动完成数据锁定工作，备份时不用关闭服务器。
  - 能刷新日志，使备份文件和日志文件的检查点能保持同步。

下面是该工具的使用示例：

```
% mysqlhotcopy db  /bakcup/              #把db数据库备份到backup/db目录里，会自动创建一
```

- 使用BACKUP TABLE语句进行备份，该语句最早出现在MySQL 3.23.25版本中，仅适用于MyISAM数据表。用法如下：

```
mysql> BACKUP TABLE mytable TO '/backup/db';        #把mytable数据表备份到/backup/db目
```

  为了执行该语句，你必须拥有那些表的FILE权限和SELECT权限，备份目录还必须是服务器可写的。该语句执行时，会先把内存中的数据写入磁盘，再把各个数据表的.frm(表结构定义文件)、.MYD(数据)文件从数据目录拷贝到备份目录。它不拷贝.MYI(索引)文件，因为它能用另外两个文件重建。BACKUP TABLE语句备份时，依次锁定数据表，当同时备份多个数据表时，数据表可能会被修改，所以备份0完成时，备份文件中的数据和现时数据表中的数据可能会有差异，为了消除该差异，我们可用只读方式锁定数据表，在备份完成后再解锁。如：

```
mysql> LOCK TABLES tb1 READ,tb2 READ;
mysql> BACKUP TABLE tb1,tb2 TO 'backup/db';
mysql> UNLOCK TABLES;
```

  使用BACKUP TABLE语句备份的数据表可用RESTORE TABLE重新加载到服务器。

- InnoDB和BDB数据库也可用mysqldump和直接拷贝法进行备份。使用直接拷贝法时应注意需把组成InnoDB和BDB数据库的所有文件都拷贝下来，如InnoDB的.frm文件、日志文件和表空间配置文件；BDB的数据文件、日志文件等。
- 使用镜像机制进行备份，我们可用SLAVE STOP语句挂起从服务器的镜像，在从服务器上通过直接拷贝法或其它工具制作备份。备份完成，用SLAVE START重新启动镜像，从服务器重新与主服务器同步，接收备份时主服务器所做的修改。

在MySQL中没有为数据库重命名的命令，但我们可用mysqldump转储数据库，再创建一个新的空数据库，把转储文件加载到该新数据库，这样就完成数据库重命名的工作。如：

```
% mysqldump old_db >db.sql            #转储db数据库数据
% mysqladmin create new_db            #新建一个空的数据库
% mysql new_db < db.sql               #把db数据库的数据加载到新的数据库中
% mysqladmin drop old_db              #删除旧的数据库
```

一个更简单的重命名数据库的方法是直接修改数据库目录名，但该方法不适用于InnoDB和BDB数据库。注意，在更名后，需在权限表中更新相关数据表信息，需执行以下语句：

```
mysql> UPDATE db SET db='new_db' WHERE db='old_db';
mysql> UPDATE tables_priv SET db='new_db' WHERE db='old_db';
mysql> UPDATE columns_priv SET db='new_db' WHERE db='old_db';
mysql> UPDATE host SET db='new_db' WHERE db='old_db';
```

6.3. 使用备份恢复数据

恢复过程包括两个信息源---备份文件和二进制日志，备份文件可使用数据恢复到执行备份时的状态，而二进制日志可恢复到发生故障时的状态。下面分别介绍如何利用这两个文件恢复一个数据库或恢复单个数据表。

恢复整个数据库的步骤：

1. 把需恢复的数据库的整个目录的内容拷贝到其它地方，以备用。

2. 使用最近的备份文件重载数据库。如果使用mysqldump生成的备份，则可使用它们作为mysql的输入重载；如果是通过拷贝数据库目录来备份的，则要关闭数据库服务器，再把备份重新拷贝到数据目录，再重启数据库服务器。

3. 通过二进制日志重做事务，恢复到出错点的数据。具体操作是这样的，用mysqlbinlog把日志转换成ASCII格式，再把它作为mysql的输入，并指定--one-database选项，以便mysql只执行你指定的数据库恢复。如：

   ```
   % ls -t -r -l binlog.[0-9]* | xargs mysqlbinlog | mysql --one-database db_name
   ```

   但上面命令只适合所有日志文件名具有相同长度的情况。否则需用下面的Perl代码来处理：

   ```
   #!/usr/bin/perl -w
   # sort_num.pl

   use strict;

   my @files = &lt;&gt;;;              #read all input file
   @files = sort { my $anum = $1 if $a =~/\.(\d+)$/;         #sort them by numeric exter
                   my $bnum = $1 if $b =~/\.(\d+)$/;
                   $anum &lt;=&gt; $bnum;
                 } @files;
   print @files;              #print them
   exit(0);

   如下使用该脚本：
   % ls -l binlog.[0-9]* | sort_num.pl | xargs mysqlbinlog | mysql --one-database db_nam
   ```

   上面讨论的是需所有日志文件的情况，但多数情况下我们只需备份后生成的几个日志文件，这时，可用以下命令来重做。

```
% mysqlbinlog binlog.1234 | mysql --one-database db_name
% mysqlbinlog binlog.1235 | mysql --one_database db_name
...
```

如果我们需恢复因执行DROP DATABASE，DROP TABLE或DELETE语句而损坏的数据库，就需在日志文件中删除该语句，否则重做后结果还是一样。所以需把日志转换成ASCII格式并保存起来，再用编辑器打开该文件，删除这些语句后再重做。

> 如果使用文本变更日志，则不需使用mysqlbinlog程序，因为该日志文件本身就是ASCII格式。

恢复使用BACKUP TABLE命令备份的数据表可用RESTORE TABLE语句：

```
备份语句：
mysql> BACKUP TABLE table_name1,table_name2 TO '/backup/table_backup';
恢复语句：
mysql> RESTORE TABLE table_name1,table_name2 FROM '/backup/table_backup';
```

恢复有外键的数据表，可用SET FOREIGN_KEY_CHECK = 0;语句先关闭键字检查，导入表后再重启它，赋值为1表示检查有效。

恢复InnoDB表空间，当服务器重启时，InnoDB处理程序会执行自动恢复工作，通过选项文件[mysqld]段中的set-variable=innodb_force_recovery=level设置自动恢复的级别，推荐典型的启动值为4。如果需从备份文件恢复，则和上面介绍的方法一样。BDB数据表的恢复也一样，启动服务器时它会进行自动恢复。

Chapter 7. MySQL程序介绍

安装完MySQL后，在MySQL的安装目录下会生成很多有用的程序，下面对这些程序进行一一介绍。

- libmysqld----嵌入式MySQL服务器，它不能独立运行，它是一个函数库，可嵌入到其它程序中，使程序具有MySQL服务器的功能。
- myisamchk和isamchk----检查和修复数据表、分析键值的分布情况、禁止或启用数据表索引的工具。
- myisampack和pack_isam----压缩数据表并生成只读数据表。
- mysql----最常用的一个与服务器交互的命令行客户端程序。
- mysqlaccess----一个用来测试数据库访问权限的脚本。
- mysqladmin----一个用来执行各种系统维护和管理工作的工具。
- mysqlbinlog----一个以ASCII格式显示二进制变更日志内容的工具。

- mysqlbug----一个用来生成程序漏洞报告的脚本。
- mysql_config----当准备编译基于MySQL的程序时，可以利用这个工具程序来确定该程序的编译选项和标志。
- mysqld----MySQL服务器程序，MySQL数据的核心。
- mysqld_multi----一个用来同时启动和关闭多个MySQL服务器的脚本。
- mysql_safe----一个用来启动和监控MySQL服务器的脚本。
- mysqldump----一个用来导出数据表内容的工具。
- mysqlhotcopy----数据库备份工具。
- mysqlimport----一个对数据表批量加载数据的工具。
- mysql_install_db----一个初始化系统权限表和数据目录的脚本。
- mysql.server----一个用来启动和关闭MySQL服务器的脚本。
- mysqlshow----一个用来显示数据库中数据表的工具。

Chapter 8. MySQL How-To

8.1. 连接数据库服务器

```
$ ./mysql -h host_name -u user_name -p
```

- `-h host_name(--host=host_name)`，连接的数据库主机名，如果在本地主机上则可省略。
- `-u user_name(--user=user_name)`，数据库用户名，在unix系统上，如果系统的登录名与数据用户名一样，则可省略。在windows系统中，可通过设置环境变量USER来给出数据库用户名，如set USER=username。
- `-p(--password)`，提供数据库用户密码，有该选项mysql就会提示你输入密码。输入的密码以星号显示，以确保安全。也可直接在-p后写上密码(-p和密码间不能有空格)，但这不安全，不推荐。

连接成功后，mysql数据库服务器会显示一些欢迎信息。接着就可用mysql>use database_name命令打开指定的数据库。

> `$ ./mysql -h host_name -u user_name -p database_name`命令可直接打开指定数据库。

8.2. 更新用户密码

```
mysql>update user set password=passowrd('your passowrd') where host='%';
```

刷新权限设置：`mysql>flush privileges;`

8.3. MySQL读取配置文件的顺序

my.cnf是MySQL数据库的配置文件，它存在多个地方，在/etc目录，数据目录和用户主目录都有。放在不同位置，里面的选项有不同的作用范围，下面是MySQL读取配置文件的顺序和作用。

```
mysql 读取配置文件的顺序:
/etc/my.cnf                Global options.
DATADIR/my.cnf             Server-specific options.
defaults-extra-file        The file specified with the --defaults-extra-file option.
~/.my.cnf                  User-specific options.
```

8.4. 重设置MySQL管理员密码的方法

有时我们会因为设置原因或时间长了忘记了数据库管理员的密码，使得我们被关在MySQL服务器外。MySQL服务器提供了一种方法可使我们在服务器上重设密码。在windows和linux/unix平台上操作稍有不同，下面分别介绍：

- Linux/Unix平台下：

  1. 用 % kill -TERM PID关闭服务器，用-TERM信息可使服务器在关闭前把内存中的数据写入磁盘。如果服务器没有响应，我们可用% kill -9 PID来强制删除进程，但不建议这样做。这时内存中的数据不会写入磁盘，造成数据不完整。如果你是用mysql_safe脚本启动MySQL服务器的，这个脚本会监控服务器的运行情况并在它被终止时重启服务器，所以如需关闭服务器，要先终止该进程，然后再真正终止mysqld进程。

  2. 使用--skip-grant-tables参数启动MySQL Server，这时MySQL服务器将不使用权限表对连接操作进行验证。你就可在不提供root密码的情况下连接上服务器，并获得root的权限。

     ```
     % mysqld_safe --skip-grant-tables &
     ```

  3. 用以下命令登录服务器，并重设密码：

     ```
      % mysql -u root         #不用密码就可连接到服务器
     mysql> use mysql
     mysql> set password for 'root'@'localhost' = password('password');
     ```

> 修改MySQL服务器帐户密码有三种方式，你可参考本笔记中数据库日常管理一章中的相关内容。在这种环境下，使用mysaladmin修改密码好象不行，还是提示要输入密码。

4. 关闭服务器，再用正常方式启动服务器。

- windows平台下：

1. 用管理员帐号登录服务器，关闭MySQL数据库服务器。
2. 使用--skip-grant-tables参数启动服务器：

```
c:\mysql\bin&gt;mysqld-nt --skip-grant-tables
```

3. 重新打开一个console窗口，用mysql命令登录服务器设置root的新密码：

```
c:\mysql\bin&gt;mysql
mysql> use mysql
mysql> set password for 'root'@'localhost' = password('password');
```

4. 关闭服务器，再用正常方式启动服务器。

8.5. NULL值

NULL是空值，代表什么也没有。它不能与值进行比较操作和算术操作，也不能和NULL进行比较，因为两个空值比较是没有意义的。我们可用“is NULL” 或 “is not NULL”来判断是不空值。如：

```
mysql> select * from test where mytest is NULL;
mysql> select * from test where mytest is not NULL;
```

在MySQL3.23以后的版本有一个新的比较操作符“<=>”，它可对NULL值进行相等比较。如：

```
mysql> select * from test where mytest <=> UNLL;
mysql> select * from test where not (mytest <=>);
```

如果查询后排序中的数据中包含NULL，则从MySQL4.0.2开始，有NULL值的数据行总是出现在查询结果的开头，即使设置的desc参数。4.0.2以前版本，如果设置了asc，则出在查询结果的开头，如果设置了desc，则出现在查询结果的结尾。

8.6. 使用SQL变量

MySQL3.23.6以后的版本允许通过查询结果来设置变量，设置好的变量可在以后使用。变量用@name定义，赋值方式用@name:=value。下面是一个在查询语句中进行赋值和使用变量的例子：

```
mysql> select @name:=id from test where mytest="test";
mysql> select * from test where mytest=@name
```

8.7. 改变默认提示符

用mysql登录进数据库后，MySQL数据的默认提示符是“mysql”，我们可设置它根据用户打开的数据库名而变化，如：

```
mysql>prompt \d>\_
none>use test
test>use mysql
mysql>
```

prompt为设置命令，\d代表当前数据库，_代表一个空格。

8.8. 非优化的全数据表DELETE操作

为了清空数据表，又需知道删除的行数和保持AUTO_INCREMENT序列的值，需用以下的删除语句：

```
# delete from table_name where 1;
```

8.9. MySQL事务处理示例

MYSQL高级特性 -- 事务处理下面以两个银行账户之间的转账为例子进行演示。要使用MySQL中的事务处理，首先需要创建使用事务表类型(如BDB = Berkeley DB或InnoDB)的表。

```
CREATE TABLE account (
account_id BIGINT UNSIGNED NOT NULL PRIMARY KEY AUTO_INCREMENT,
balance DOUBLE
) TYPE = InnoDB;
```

要在事务表上使用事务处理，必须要首先关闭自动提交：

```
SET AUTOCOMMIT = 0;
```

事务处理以BEGIN命令开始：

```
BEGIN;
```

现在mysql客户处在于服务器相关的事物上下文中。任何对事务表所做的改变在提交之前不会成为永久性的改变。

```
UPDATE ACCOUNT SET balance = 50.25 WHERE account_id = 1;
UPDATE ACCOUNT SET balance = 100.25 WHERE account_id = 2;
```

在做出所有的改变之后，使用COMMIT命令完成事务处理：

```
COMMIT;
```

当然，事务处理的眞正优点是在执行第二条语句发生错误时体现出来的，若在提交前终止整个事务，可以进行回滚操作：

```
ROLLBACK;
```

下面是另一个例子，通过MYSQL直接进行数学运算：

```
SELECT @first := balance FROM account WHERE account_id = 1;
SELECT @second := balance FROM account WHERE account_id = 2;
UPDATE account SET balance = @first - 25.00 WHERE account_id = 1;
UPDATE account SET balance = @second + 25.00 WHERE account_id = 2;
```

除了COMMIT命令外，下列命令也会自动结束当前事务：

```
ALTER TABLE
BEGIN
CREATE INDEX
DROP DATABASE
DROP TABLE
LOCK TABLES
RENAME TABLE
TRUNCATE
UNLOCK TABLES
```

Python 学习笔记 基础篇

整理：Jims of 肥肥世家

jims.yang@gmail.com

发布时间：2004年07月10日

更新时间：2006年06月14日，把参考篇的内容合并进来。

Abstract

现时国内python的中文资料极少，使学习Python较困难。国外的资料虽多，但都是英文的，使我们学习起来很不方便。有鉴于此，我开始了Python中文资料库的整理工作，以推动Python的发展和在中国的应用。在自由的世界里，正因为有你的支持和帮助，才使我得以不断前进。我相信我们每人一小步就可带动python在中国前进一大步。

Table of Contents

Chapter 1. 绪论

1.1. Python历史

Python是一种开源的面向对象的脚本语言，它起源于1989年末，当时，CWI（阿姆斯特丹国家数学和计算机科学研究所）的研究员Guido van Rossum需要一种高级脚本编程语言，为其研究小组的Amoeba分布式操作系统执行管理任务。为创建新语言，他从高级数学语言ABC（ALL BASIC CODE）汲取了大量语法，并从系统编程语言Modula-3借鉴了错语处理机制。Van Rossum把这种新的语言命名为Python（大蟒蛇）---来源于BBC当时正在热播的喜剧连续剧“Monty Python”。

Python于1991年初公开发行，由于功能强大和采用开源方式发行，Python的发展得很快，用户越来越多，形成了一个强大的社区力量。2001年，Python的核心开发团队移师Digital Creations公司，该公司是Zope（一个用Python编写的web应用服务器）的创始者。现在最新的版本是python2.3.4，大家可到http://www.python.org上了解最新的Python动态和资料 。

1.2. Python功能简介

Python是一种解析性的，交互式的，面向对象的编程语言，类似于Perl、Tcl、Scheme或Java。

Python一些主要功能介绍:

- Python使用一种优雅的语法，可读性强。
- Python是一种很灵活的语言，能帮你轻松完成编程工作。并可作为一种原型开发语言，加快大型程序的开发速度。
- 有多种数据类型：numbers (integers, floating point, complex, and unlimited-length long integers), strings (ASCII 和 Unicode), lists, dictionaries。
- Python支持类和多层继承等的面向对象编程技术。
- 代码能打包成模块和包，方便管理和发布。
- 支持异常处理，能有效捕获和处理程序中发生的错误。
- 强大的动态数据类型支持，不同数据类型相加会引发一个异常。
- Python支持如生成器和列表嵌套等高级编程功能。
- 自动内存碎片管理，有效利用内存资源。
- 强大的类库支持，使编写文件处理、正则表达式，网络连接等程序变得相当容易。
- Python的交互命令行模块能方便地进行小代码调试和学习。
- Python易于扩展，可以通过C或C++编写的模块进行功能扩展。
- Python解析器可作为一个编程接口嵌入一个应用程序中。
- Python可运行在多种计算机平台和操作系统中，如各位unix，windows，MacOS,OS/2等等。
- Python是开源的，可自由免费使用和发布，并且可用于商业用途以获取利润。如想详细了解Python的许可协议可到以下网址查询http://www.python.org/psf/license.html

1.3. 应用范围

- 系统编程，提供大量系统接口API，能方便进行系统维护和管理。
- 图形处理，有PIL、Tkinter等图形库支持，能方便进行图形处理。
- 数学处理，NumPy扩展提供大量与许多标准数学库的接口，
- 文本处理，python提供的re模块能支持正则表达式，还提供SGML，XML分析模块，许多程序员利用python进行XML程序的开发。
- 数据库编程，程序员可通过遵循Python DB-API（数据库应用程序编程接口）规范的模块与Microsoft SQL Server，Oracle，Sybase，DB2，Mysql等数据库通信。python自带有一个Gadfly模块，提供了一个完整的SQL环境。
- 网络编程，提供丰富的模块支持sockets编程，能方便快速地开发分布式应用程序。
- 作为Web应用的开发语言，支持最新的XML技术。
- 多媒体应用，Python的PyOpenGL模块封装了“OpenGL应用程序编程接口”，能进行二维和三维图像处理。PyGame模块可用于编写游戏软件。

1.4. 如何开始？

- 进入交互命令行方式。如果是linux类的系统，python解析器应该已经安装在/usr/local/bin/python中，直接打python就可进入交互式命令行界面，如下所示:

  ```
  Python 2.3.3 (#1, Apr 27 2004, 15:17:58)
  [GCC 3.2 20020903 (Red Hat Linux 8.0 3.2-7)] on linux2
  Type "help", "copyright", "credits" or "license" for more information.
  >>>
  ```

  “>>>”符号是Python命令行界面的提示符，可按CTRL+D退出，如果是windows环境的话就要按CTRL+Z了。还可以用以下命令退出命令行界面：“import sys；sys.exit()”。如果是windows系统，可到http://www.python.org/download/下载最新的安装程序进行安装。安装完成后直接打python也可进入命令行界面。命令行是python最简单直观，也是最方便的一种执行环境，我们可以在这里学习python语法和调试程序。如果要打印"hello world"可以输入以下命令：

  ```
  >>>print "hello world"
  hello world
  ```

- 以模块文件方式运行。模块文件是包含python语句的文本，以.py结尾。运行模块文件只要输入python xxx.py就可以了。
- 以linux脚本方式运行。和shell脚本差不多，以vi或其它文本编辑器输入以下内容:

```
#!/usr/local/bin/python
print "test ..........."
```

存盘后，把文件属性改为可执行，就可象shell脚本一样执行了。

- **Table 1.1. Python**命令行选项

```
| 选项 | 作用 |
| --- | --- |
| -c cmd | 在命令行直接执行python代码。如python -c 'print "hello world"'。 |
| -d | 脚本编译后从解释器产生调试信息。同PYTHONDEBUG=1。 |
| -E | 忽略环境变量。 |
| -h | 显示python命令行选项帮助信息。 |
| -i | 脚本执行后马上进入交互命令行模式。同PYTHONINSPECT=1。 |
| -O | 在执行前对解释器产生的字节码进行优化。同 PYTHONOPTIMIZE=1。 |
| -OO | 在执行前对解释器产生的字节码进行优化，并删除优化代码中的嵌入式文档字符串。 |
| -Q arg | 除法规则选项，-Qold(default)，-Qwarn，-Qwarnall，-Qnew。 |
| -S | 解释器不自动导入site.py模块。 |
| -t | 当脚本的tab缩排格式不一致时产生警告。 |
| -u | 不缓冲stdin、stdout和stderr，默认是缓冲的。同PYTHONUNBUFFERED=1。 |
| -v | 产生每个模块的信息。如果两个-v选项，则产生更详细的信息。同PYTHONVERBOSE=x。 |
| -V | 显示Python的版本信息。 |
| -W arg | 出错信息控制。(arg is action:message:category:module:lineno) |
| -x | 忽略源文件的首行。要在多平台上执行脚本时有用。 |
| file | 执行file里的代码。 |
| - | 从stdin里读取执行代码。 |
```

Chapter 2. Python编程习惯与特点

2.1. 代码风格

- 在Python中，每行程序以换行符代表结束，如果一行程序太长的话，可以用“\”符号扩展到下一行。在python中以三引号(""")括起来的字符串，列表，元组和字典都能跨行使用。并且以小括号(...)、中括号[...]和大括号{...}包围的代码不用加“\”符也可扩展到多行。如：
- 在Python中是以缩进来区分程序功能块的，缩进的长度不受限制，但就一个功能块来讲，最好保持一致的缩进量。
- 如果一行中有多条语句，语句间要以分号（;）分隔。
- 以“#”号开头的内容为注释，python解释器会忽略该行内容。
- 在python中，所有标识符可以包括英文、数字以及下划线（_），但不能以数字开头。python中的标识符是区分大小写的。
- 以下划线开头的标识符是有特殊意义的。以单下划线开头（_foo）的代表不能直接访问的类属性，需通过类提供的接口进行访问，不能用“from xxx import *”而导入；以双下划线开头的（__foo）代表类的私有成员；以双下划线开头和结尾的（__foo__）代表python里特殊方法专用的标识，如__init__（）代表类的构造函数。

- 在交互模式下运行python时，一个下划线字符(_)是特殊标识符，它保留了表达式的最后一个计算结果。

```
>>> "hello"
'hello'
>>> _
'hello'
>>> 10+10
20
>>> _
20
```

- 在python中，函数、类、模块定义的第一段代码如果是字符串的话，就把它叫作文件字串，可通过__doc__属性访问。如:

```
def test():
"this is a document string"

return 100+1000

>>>print test.__doc__
this is a document string
```

2.2. 保留字

```
and         elif        global        or          yield
assert        else        if         pass
break         except        import         print
class         exec         in         raise
continue    finally        is         return
def         for         lambda         try
del         from         not         while
```

2.3. Python运算符和表达式

2.3.1. Python运算符

Table 2.1. Python运算符列表

运算符	描述
x+y，x-y	加、减，“+”号可重载为连接符
x*y，x**y，x/y，x%y	相乘、求平方、相除、求余，“*”号可重载为重复，“%”号可重载为格式化
<，<=，>，>=，==，<>，!=	比较运算符
+=，-=，*=，/=，%=，**=，<<=，>>=，&=，^=，|=	自变运算符
x|y	按位或
x^y	按位异或
x&y	按位与
~x	按位取反
x<<，x>>y	x向左或向右移y位
is, is not	等同测试
in, not in	是否为成员测试
or，and，not	逻辑运算符
x[i]，x[i:j]，x.y，x(...)	索引，分片，限定引用，函数调用
(...)，[...]，{...}，'...'	元组，列表，字典，转化为字符串

2.3.2. 运算符优先顺序

Table 2.2. 运算符优先顺序列表**(**从最高到最低**)**

运算符	描述
'expr'	字符串转换
{key:expr,...}	字典
[expr1,expr2...]	列表
(expr1,expr2,...)	元组
function(expr,...)	函数调用
x[index:index]	切片
x[index]	下标索引取值
x.attribute	属性引用
~x	按位取反
+x，-x	正，负
x**y	幂
x*y，x/y，x%y	乘，除，取模
x+y，x-y	加，减
x<<y，x>>y	移位
x&y	按位与
x^y	按位异或
x|y	按位或
x<y，x<=y，x==y，x!=y，x>=y，x>y	比较
x is y，x is not y	等同测试
x in y，x not in y	成员判断
not x	逻辑否
x and y	逻辑与
x or y	逻辑或
lambda arg,...:expr	Lambda匿名函数

2.3.3. 真值表

Table 2.3.

对象/常量	值
""	假
"string"	眞
0	假
>=1	眞
<=-1	眞
()空元组	假
[]空列表	假
{}空字典	假
None	假

2.3.4. 复合表达式

- 对于and，当计算a and b时，python会计算a，如果a为假，则取a值，如果a为眞，则python会计算b且整个表达式会取b值。如：

```
>>> a,b=10,20
>>> a and b   #a is true
20
>>> a,b=0,5   #a is false
>>> a and b
0
```

- 对于or，当计算a or b时，python会计算a，如果a为眞，则整个表达式取a值，如果a为假，表达式将取b值。如：

```
>>> a,b=10,20
>>> a or b
10
>>> a,b=0,5
>>> a or b
5
```

- 对于not，not将反转表表达式的“实际值”，如果表达式为眞，not为返回假，如为表达式为假，not为返回眞。如：

```
>>> not 2
False
>>> not 0
True
>>> not "test"
False
>>> not ""
True
```

2.4. 给变量赋值

- 简单赋值，Variable(变量)=Value(值)。

```
>>>a=1
>>>b=2
>>>print a,b
1 2
```

- 多变量赋值，Variable1,variable2,...=Value1,Value2,...

```
>>>a,b,c=1,2,3
>>>print a
1
>>>print b
2
>>>print c
3
```

  多变量赋值也可用于变量交换，接上例：

```
>>>a,b,c=c,b,a
>>>print a
3
>>>print b
2
>>>print c
1
```

- 多目标赋值，a=b=variable

```
>>> a=b=1
>>> a
1
>>> b
1
>>> a=2
>>> a
2
>>> b
1
```

- 自变赋值，如+=，-=，*=等。在自变赋值中，python仅计算一次，而普通写法需计算两次；自变赋值会修改原始对象，而不是创建一个新对象。

Chapter 3. Python内建对象类型

在Python中，所有数据都是对象，数据有各种类型，如数值型、列表型、字符串型等。除系统内建的数据类型外，程序员也可以创建自已的数据类型。以下主要介绍Python内建的数据类型。

3.1. Number数值型

在python中，数值有四种类型，分别是整型、长整形、浮点型和复数。

- 整型---从-2147483648至2147483647，有符号位32位长，可表达的最大数为2^31-1。如：number=123，number1=-123。在数字前加0x或0X 前缀表示十六进制数，在数字前加前缀0表示八进制数，与C/C++ and perl一样。

  > 为方便起见，sys模块包含一个maxint成员，该成员保留了整形变量的最大正数值。

  ```
  >>> import sys
  >>> print sys.maxint
  2147483647
  ```

- 长整型---python支持任意长度的长整型，长整型的最大值和最小值由可用的内存确定。长整型数在数字常量尾加L or l，一般都是用L，因为小写的l太容易与数字1混淆了。如：long=1232132131231232132132131L。
- 浮点数---python支持普通十进制和科学计数法表示的浮点数。如：number=123.456，nubmer1=123.2E10。浮点数在python中的存储格式与C中的双精度数相同。
- 复数---复数的实部和虚部用加号分开，虚部使用后缀j表示，如：number=1.2+2j

3.2. String字符串型

- 字符串在python被看成是单个字符的序列，具有序列对象的特殊功能，字符串是固定的，不可变的。如：string="hello world"。
- 可在字符串中使用单引号和双引号。如：string="I'm a boy"。
- 字符串内部的一个反斜杠“\”可允许把字符串放于多行：如：

  ```
  >>> "test \
  ... python"
  'test python'
  ```

- 使用三个单引号或双引号可使字符串跨行显示。如：

  ```
  helptext="""this a help test.if you have any quesions.
              please call me anytime.I will help you.I
              like python.I hope so as you."""
  ```

- 使用“+”号可连接字符串。如：string = "hello" + "world"，注意，不能将字符串与其它对象进行连接。如string = "ok" + 5。其实不用“+”号，直接用空格也可连接两个字符串。如：string="hello" "world"。

- 可用“*”号重复字符串，如：'hello'*5会生成'hellohellohellohellohello'。
- 可用索引访问字符串中的字符。如：string="hello world"，print string[1]将显示字符e。
- 字符串可用in或not in运算符来测试字符是不属于一个字符串的成员。
- 可对字符串分片，如string="hello world",print string[6:]将显示world。分片的格式为：

  ```
  string[start:end]
  ```

  分片和索引的规则如下：

  - 返回的字符串包含从start起始到end但不包括end结束的所有字符。
  - 若指定了start但未指定end，则一直向后分片，直至字符串结束。
  - 若指定了end但未指定start，则从0开始分片直至end，但不包括end指定的字符。
  - 若start和end为负数，则索引从字符串尾部开始算起，最后一个字符为-1。

python提供了一个string模块来进行字符串处理。

3.2.1. 字符串的格式化

象C 中的sprintf函数一样，可以用“%”来格式化字符串。

Table 3.1. 字符串格式化代码

格式	描述
%%	百分号标记
%c	字符及其ASCII码
%s	字符串
%d	有符号整数(十进制)
%u	无符号整数(十进制)
%o	无符号整数(八进制)
%x	无符号整数(十六进制)
%X	无符号整数(十六进制大写字符)
%e	浮点数字(科学计数法)
%E	浮点数字(科学计数法，用E代替e)
%f	浮点数字(用小数点符号)
%g	浮点数字(根据值的大小采用%e或%f)
%G	浮点数字(类似于%g)
%p	指针(用十六进制打印值的内存地址)
%n	存储输出字符的数量放进参数列表的下一个变量中

> %格式化符也可用于字典，可用%(name)引用字典中的元素进行格式化输出。
>
> 负号指时数字应该是左对齐的，“0”告诉Python用前导0填充数字，正号指时数字总是显示它的正负(+，-)符号，即使数字是正数也不例外。
>
> 可指定最小的字段宽度，如："%5d" % 2。也可用句点符指定附加的精度，如："%.3d" % 3。

3.2.2. 转义字符

在需要在字符中使用特殊字符时，python用反斜杠()转义字符。如下表：

Table 3.2. python支持的转义字符表

转义字符	描述
(在行尾时)	续行符
\	反斜杠符号
\'	单引号
\"	双引号
\a	响铃
\b	退格(Backspace)
\e	转义
\000	空
\n	换行
\v	纵向制表符
\t	横向制表符
\r	回车
\f	换页
\oyy	八进制数yy代表的字符，例如：\o12代表换行
\xyy	十进制数yy代表的字符，例如：\x0a代表换行
\other	其它的字符以普通格式输出

3.2.3. Unicode字符串

在python2.0中才完全支持Unicode字符串，Unicode字符采用16位(0---65535)值表示，能进行多语言支持。要使用Unicode字符串，只要在字符串前加上“u”即可。如：

```
>>> a=u"test"
>>> print a
test
```

原始Unicode字符串用ur前缀，如：

```
>>> u'hello world\0020'
u'hello world\x020'
>>> ur'hello world\0020'
u'hello world\\0020'
```

3.2.3.1. Unicode转换

只要和Unicode连接，就会产生Unicode字串。如：

```
>>> 'help'
'help'
>>> 'help，' + u'python'
u'help，python'
```

对于ASCII(7位)兼容的字串，可和内置的str()函数把Unicode字串转换成ASCII字串。如：

```
>>> str(u'hello world')
'hello world'
```

> 转换非ASCII兼容的字串会出错。编码和译码字符串时的错误引发UnicodeError异常。

可使用encode()函数转换Unicode字串格式：

```
u'unicode\xb1\xe0\xc2\xeb\xb2\xe2\xca\xd4'
>>> a.encode('utf-8')   #转换成utf-8，显示结果会根据终端的字符集支持不同而不同，下面是在GB18030下的
'unicode\xc2\xb1\xc3\xa0\xc3\x82\xc3\xab\xc2\xb2\xc3\xa2\xc3\x8a\xc3\x94'
```

可使用unicode()函数把字符串转换成unicode格式，如：

```
>>> a=u'unicode测试'
>>> a
u'unicode\xb2\xe2\xca\xd4'
>>> a.encode('utf-8')     #把unicode字串转换成utf-8
'unicode\xc2\xb2\xc3\xa2\xc3\x8a\xc3\x94'
>>> b=a.encode('utf-8')   #给变量b赋值
>>> b
'unicode\xc2\xb2\xc3\xa2\xc3\x8a\xc3\x94'
>>>unicode(b,'utf-8')           #用unicode()函数把utf-8格式字串转换回unicode格式。
u'unicode\xb2\xe2\xca\xd4'      #和原始的这是a相同
```

ord()支持unicode，可以显示特定字符的unicode号码，如：

```
>>>ord('A')
65
```

使用unichr()函数可将unicode号码转换回unicode字符，如：

```
>>> unichr(65)
u'A'
```

3.2.4. 原始字符串

有时我们并不想让转义字符生效，我们只想显示字符串原来的意思，这就要用r和R来定义原始字符串。如：

```
print r'\t\r'
```

实际输出为“\t\r”。

3.3. List列表

- 列表是序列对象，可包含任意的Python数据信息，如字符串、数字、列表、元组等。列表的数据是可变的，我们可通过对象方法对列表中的数据进行增加、修改、删除等操作。可以通过list(seq)函数把一个序列类型转换成一个列表。列表的几个例子：
  - `list = [ "a", "b", "c" ]`，这是字符列表。
  - `list = [ 1, 2, 3, 4 ]`，这是数字列表。
  - `list = [ [1,2,3,4], ["a","b","c"] ]`，这是列表的列表。
  - `list = [ (1,2,3,4), ("a","b","c") ]`，这是元组列表。
  - list((1,2))把一个元组转换成一个列表[1,2]，list('test')可把字符串转换成['t','e','s','t']列表。
- 访问列表可通过索引来引用，如：list[0]将引用列表的第一个值。list[0:1]返回第一和第二个元素。
- 用range()和xrange()函数可自动生成列表，具体用法请参考“python参考篇”的内容。
- 可通过列表综合来创建列表，该功能是在python2.0版本中新增加的。如果想对列表中的每个项进行运算并把结果存储在一个新列表中，可者想创建一个仅包含特定满足某种条件的项，采用该方法是很适合的。如：[x*x for x in range(1,10)]会得到一个X的平方的新列表；我们还可添加if条件控制输出，如：[x*x for x in range(1,10) if x%2==0]；还可在列表中使用多个for语句，如：

  ```
  >>> [x+y for x in "123" for y in "abc"]
  ['1a', '1b', '1c', '2a', '2b', '2c', '3a', '3b', '3c']
  ```

  x,y值可取列表或元组等，以构成更复杂的结构。
- “+”号可连接两个列表。
- 访问列表的列表(嵌套列表)可用list[1][0]，这将访问嵌套中的第二个列表的第一个元素。
- 可用数字与列表相乘以复制内容，如：list*2会得到一个[1,2,3,4,1,2,3,4]的列表。注意，不能用列表与列表相乘。
- 由于列表是可变的，我们可用赋值语句进行操作，如：list[0] = 2。
- 列表对象方法可对列表进行操作，如列表内容的添加，删除，排序等。如list.sort()可对list列表进行排序。

````
**Table 3.3\. 列表对象支持的方法**

```
| 方法 | 描述 |
| --- | --- |
| append(x) | 在列表尾部追加单个对象x。使用多个参数会引起异常。 |
| count(x) | 返回对象x在列表中出现的次数。 |
| extend(L) | 将列表L中的表项添加到列表中。返回None。 |
| Index(x) | 返回列表中匹配对象x的第一个列表项的索引。无匹配元素时产生异常。 |
| insert(i,x) | 在索引为i的元素前插入对象x。如list.insert(0,x)在第一项前插入对象。返回None。 |
| pop(x) | 删除列表中索引为x的表项，并返回该表项的值。若未指定索引，pop返回列表最后一项。 |
| remove(x) | 删除列表中匹配对象x的第一个元素。匹配元素时产生异常。返回None。 |
| reverse() | 颠倒列表元素的顺序。 |
| sort() | 对列表排序，返回none。bisect模块可用于排序列表项的添加和删除。 |
```
````

3.4. Tuple元组

Tuple(元组)和List(列表)很相似，但元组是不可变的。不能对元组中的元素进行添加，修改和删除操作。如果需修改元组内容只有重建元组。元组用小括号来表示。如tuple=(1,2,3)。

- tuple=(1,)，这是单个元素的元组表示，需加额外的逗号。
- tuple=1，2，3，4，这也可以是一个元组，在不使用圆括号而不会导致混淆时，Python允许不使用圆括号的元组。
- 和列表一样，可对元组进行索引、分片、连接和重复。也可用len()求元组长度。

  > 元组的索引用tuple[i]的形式，而不是tuple(i)。

- 和列表类似，使用tuple(seq)可把其它序列类型转换成元组。

3.5. 序列对象

上面介绍的字符串、列表和元组的对象类型均属于称为序列的Python对象。它是一种可使用数字化索引进行访问其中元素的对象。

- 可用算术运算符联接或重复序列。
- 比较运算符(<，<=，>，>=，!=，==)也可用于序列。
- 可通过下标(test[1])，切片(test[1:3])和解包来访问序列的某部份。解包示例如下：

  ```
  >>>s=1,2,3
  >>>x,y,z=s
  >>>print x,y,z
  1,2,3
  ```

- in运算符可判断当有对象是否序列对象成员，如:

```
>>>list = [1,2,3]
>>>1 in list
1
>>>4 in list
0
```

- 也可通过循环运算符对序列对象进行迭代操作。如:

```
for day in days:
    print day
```

有关序列的处理函数请参考“python参考篇”相关内容，这里就不详细讲了。

3.6. Dictionary字典

字典是一个用大括号括起来的键值对，字典元素分为两部份，键(key)和值。字典是python中唯一内置映射数据类型。通过指定的键从字典访问值。如：

```
monthdays = { "Jan":31, "Feb":28, "Mar":31, "Apr":30, "May":31, "Jun":30, "Jul":31, "Aug"
```

- 字典可嵌套，可以在一个字典里包含另一个字典。如test={"test":{"mytest":10} }
- 可用键访问字典，如monthdays["Jan"]，可访问值31。如果没有找到指定的键，则解释器会引起异常。
- 字典是可修改，如monthdays["Jan"]=30，可把Jan的值由31改为30。如monthdays["test"]=30可添加一个新键值对。
- del monthdays["test"]可删除字典条目。
- 字典不属序列对象，所以不能进行连接和相乘操作。字典是没有顺序的。
- 字典提供keys和values方法，用来返回字典中定义的所有键和值。
- 和列表一样，字典也提供了对象方法来对字典进行操作。

````
**Table 3.4\. 字典方法**

```
| 方法 | 描述 |
| --- | --- |
| has_key(x) | 如果字典中有键x，则返回眞。 |
| keys() | 返回字典中键的列表 |
| values() | 返回字典中值的列表。 |
| items() | 返回tuples的列表。每个tuple由字典的键和相应值组成。 |
| clear() | 删除字典的所有条目。 |
| copy() | 返回字典高层结构的一个拷贝，但不复制嵌入结构，而只复制对那些结构的引用。 |
| update(x) | 用字典x中的键值对更新字典内容。 |
| get(x[,y]) | 返回键x，若未找到该键返回none，若提供y，则未找到x时返回y。 |
```
````

3.7. File文件

可用内置的open()函数对文件进行操作。如：

```
input = open("test.txt")
for line in input.readlines():
    print line
input.close()
```

3.8. 理解引用

- Python把一块数据存储在对象中，变量是对象的唯一引用；它们是计算机内存中特殊地点的名字。所有对象都具有唯一的身份号、类型和值。对象的类型不会改变，对于可变类型而言，它的值是可变的。id(obj)函数可用于检索对象的身份，也就是内存中的对象的地址。
- 每个对象都包含引用计数器，它记录当前有多少个变量正在引用该对象。当给对象指定一个变量或使对象成为列表或其它包容器的成员时，引用计数就增加；当从包容器中撤消、重新分配或删除对象时，引用计数减少。当引用计数达到0值时(即没有任何变量引用这个对象)，python的回收机制会自动回收它使用的内存。注意，del可用来删除变量，但不能删除对象。

  > sys.gettrefcount(obj)函数可返回给定对象的引用计数。

3.9. copy and deepcopy

通过给列表分配一个变量能创建对列表的引用，如果要创建列表的副本就要理解浅副本和深副本的概念。

- 列表或其他包容器对象的浅副本(Shallow)能够生成对象本身的副本，但也会创建对由列表包含的对象的引用。可用分片(object[:])和copy模块的copy(obj)函数创建。

- 列表或其他对象包容器对象的深副本能够生成对象本身的副本，并递归地生成所有子对象的副本。可用copy模块的deepcopy(obj)函数创建。

比较两种副本，一般情况下表现一样，但当列表内包含另一个列表的情况下，父列表的浅副本将包含对子列表引用，而不是独立副本。其结果是，当更改内部列表时，从父列表的两个副本中都可见，如：

```
>>> a=[1,2,3,[4,5]]
>>> b=a[:]
>>> b
[1, 2, 3, [4, 5]]
>>> a[3].remove(4)
>>> a
[1, 2, 3, [5]]
>>> b
[1, 2, 3, [5]]
```

如果是深副本，就不会出现这种情况。如：

```
>>> a=[1,2,3,[4,5]]
>>> b=copy.deepcopy(a)
>>> b
[1, 2, 3, [4, 5]]
>>> a[3].remove(4)
>>> a
[1, 2, 3, [5]]
>>> b
[1, 2, 3, [4, 5]]
```

3.10. 标识数据类型

可通过type(obj)函数标识数据类型，如：

```
>>> type(a)
<type 'list'>
>>> type(copy)
<type 'module'>
>>> type(1)
<type 'int'>
```

types模块包含Python的内置数据类型的类型对象。如：

```
>>> import types
>>> types.ListType
<type 'list'>
>>> types.IntType
<type 'int'>
```

3.11. 数组对象

数组对象与列表类似，但数组只包含某些类型的简单数据。所以当数据较简单，且要求性能好的情况下，使用数组是一个好的选择。

Table 3.5. 数组类型代码

代码	等价的C类型	以字节为单位的最小尺寸
c	char	1
b(B)	byte(unsigned byte)	1
h(H)	short(unsigned short)	2
i(I)	int(unsigned int)	2
l(L)	long(unsigned long)	4
f	float	4
d	double	8

数组创建方法如下：

```
>>> import array
>>> z=array.array("b")
>>> z.append(1)
>>> z
array('b', [1])
```

数组的itemsize和typecode成员可分别检索数组项的大小和数组对象的类型代码，如：

```
>>> z.itemsize
1
>>> z.typecode
'b'
```

3.1. 数组类型与其它数据类型的转换

- tolist()方法可把数组转换为列表，如：

  ```
  >>> z.tolist()
  [1, 2, 3]
  ```

  fromlist(list)方法可把列表项附加到数组的末尾，如：

  ```
  >>> z.fromlist([10,11])
  >>> z
  array('b', [1, 2, 3, 10, 11])
  ```

  > 如添加的列表类型与数组类型不同，则fromlist(list)不会把任何项添加到数组对象中。

- tostring()方法，可以把数组转换为字节的序列，如：

```
>>> z.tostring()
'\x01\x02\x03\n\x0b'
```

fromstring(list)方法刚好与tostring()相反，它获取一个字节串，并把它们转换为数组的值。如：

```
>>> z.fromstring("\x0b")
>>> z
array('b', [1, 2, 3, 10, 11, 11])
```

- tofile(file)方法可把数组转换为字节的序列，并把它们写入文件，如：

```
>>> f=open("aa","wb")
>>> z.tofile(f)
>>> f.close()
```

fromfile(file,count)方法用于从文件对象中读取特定数目的项，并把它们附加到数组中，如：

```
>>> z.fromfile(open("aa","rb"),2)
>>> z
array('b', [1, 2, 3, 10, 11, 11, 1, 2])
```

当取数项大于文件数据项时，formfile会产生EOFError异常。

- 数组对象支持列表中的很多相同函数和方法：len，append等。访问成员的方法也可列表一样，可用下标和分片。

Chapter 4. 控制语句

流程控制是程序设计中一个重要的内容，Python支持三种不同的控制结构：if，for和while。

- if语句判断表达式是否为真，如果为真则执行指定语句。if语句的格式如下：

```
if   EXPRESSION1:
         STATEMENT1
elif EXPRESSION2:
         STATEMENT2
else:
         STATEMENT3
```

如果第一个表达式为真，则执行statement1，否则进行进一步的测试，如果第二个表达式为真则执行statement2，否则执行statement3。

> 注意语句的缩进量要保持一致。在python中没有switch和case语句，我们可通过多重elif来达到相同的效果。

示例：

```
#!/usr/bin/env python

mytest = raw_input("please input a number:")
mytest = int(mytest)
if mytest == 10:
        print "you input number is ten."
elif mytest == 20:
        print "you input number is twenty."
else:
        print "another number."
```

脚本的执行效果：

```
t03:~# python test.py
please input a number:10
you input number is ten.
t03:~# python test.py
please input a number:20
you input number is twenty.
t03:~# python test.py
please input a number:777
another number.
```

- while进行循环控制，它对表达式进行测试，如果为真，则循环执行循环体。格式如下：

```
while EXPRESSION:
          STATEMENT
else:
          STATEMENT
```

如果测试为假，则会执行else块。如果循环被中断(break)，则else块不会执行。

示例：

```
>>> a = 0
>>> while a &gt; 5:
...     a = a + 1
...     print a
... else:
...     print "a's value is five"
...
1
2
3
4
5
a's value is five
```

- for循环可遍历对象，并可进行迭代操作。语名格式如下：

```
for TARGET in OBJECTS：
       STATEMENT
else:
       STATEMENT
```

和while一样，在循环正常退出时，会执行else块。

示例：

```
>>> mylist = "for statement"
>>> for word in mylist:
...     print word
... else:
...     print "End list"
...
f
o
r

s
t
a
t
e
m
e
n
t
End list
```

- 在循环的过程中，我们可使用循环控制语句来控制循环的执行。有三个控制语句，分别是break、continue和pass。它们的作用分别是：
  - break语句会立即退出当前循环，不会执行else块的内容。

    示例：

    ```
    >>> mylist = ["zope","python","perl","Linux"]
    >>> for technic in mylist:
    ...     if technic == "perl":
    ...             break
    ...     else:
    ...             print technic
    ...
    zope
    python
    ```

  - continue语句会忽略后面的语句，强制进入下一次循环。

    示例：

```
>>> mylist = ["zope","python","perl","Linux"]
>>> for technic in mylist:
...     if technic == "perl":
...             continue
...     else:
...             print technic
...
zope
python
Linux
```

- pass不做任何事情。

  示例：

```
>>> for technic in mylist:
...     if technic == "perl":
...             pass
...     else:
...             print technic
...
zope
python
Linux
```

Chapter 5. 函数

函数是一个能完成特定功能的代码块，可在程序中重复使用，减少程序的代码量和提高程序的执行效率。在python中函数定义语法如下：

```
def function_name(arg1,arg2[,...]):
    statement
[return value]
```

> 返回值不是必须的，如果没有return语句，则Python默认返回值None。

函数名的命名规则：

- 函数名必须以下划线或字母开头，可以包含任意字母、数字或下划线的组合。不能使用任何的标点符号；
- 函数名是区分大小写的。
- 函数名不能是保留字。

Python使用名称空间的概念存储对象，这个名称空间就是对象作用的区域， 不同对象存在于不同的作用域。下面是不同对象的作用域规则：

- 每个模块都有自己的全局作用域。
- 函数定义的对象属局部作用域，只在函数内有效，不会影响全局作用域中的对象。

- 赋值对象属局部作用域，除非使用global关键字进行声明。

LGB规则是Python查找名字的规则，下面是LGB规则：

- 大多数名字引用在三个作用域中查找：先局部(Local)，次之全局(Global)，再次之内置(Build-in)。

```
>>> a=2
>>> b=2
>>> def test(b):
...     test=a*b
...     return test
>>>print test(10)
20
```

  b在局部作用域中找到,a在全局作用域中找到。

- 如想在局部作用域中改变全局作用域的对象，必须使用global关键字。

```
#没用global时的情况
>>> name="Jims"
>>> def set():
...     name="ringkee"
...
>>> set()
>>> print name
Jims
#使用global后的情况
>>> name="Jims"
>>> def set1():
...     global name
...     name="ringkee"
...
>>> set1()
>>> print name
ringkee
```

- 'global'声明把赋值的名字映射到一个包含它的模块的作用域中。

函数的参数是函数与外部沟通的桥梁，它可接收外部传递过来的值。参数传递的规则如下：

- 在一个函数中对参数名赋值不影响调用者。

```
>>> a=1
>>> def test(a):
...     a=a+1
...     print a
...
>>> test(a)
2
>>> a
1              # a值不变
```

- 在一个函数中改变一个可变的对象参数会影响调用者。

```
>>> a=1
>>> b=[1,2]
>>> def test(a,b):
...     a=5
...     b[0]=4
...     print a,b
...
>>> test(a,b)
5 [4, 2]
>>> a
1
>>> b
[4, 2]    # b值已被更改
```

参数是对象指针，无需定义传递的对象类型。如：

```
>>> def test(a,b):
...     return a+b
...
>>> test(1,2)   #数值型
3
>>> test("a","b")   #字符型
'ab'
>>> test([12],[11])   #列表
[12, 11]
```

函数中的参数接收传递的值，参数可分默认参数，如：

```
def function(ARG=VALUE)
```

元组（Tuples）参数：

```
def function(*ARG)
```

字典（dictionary）参数：

```
def function(**ARG)
```

一些函数规则：

- 默认值必须在非默认参数之后；
- 在单个函数定义中，只能使用一个tuple参数（*ARG）和一个字典参数（**ARG）。
- tuple参数必须在连接参数和默认参数之后。
- 字典参数必须在最后定义。

5.1. 常用函数

- abs(x)

abs()返回一个数字的绝对值。如果给出复数，返回值就是该复数的模。

```
>>>print abs(-100)
100
>>>print abs(1+2j)
2.2360679775
```

- callable(object)

callable()函数用于测试对象是否可调用，如果可以则返回1(真)；否则返回0(假)。可调用对象包括函数、方法、代码对象、类和已经定义了“调用”方法的类实例。

```
>>> a="123"
>>> print callable(a)
0
>>> print callable(chr)
1
```

- cmp(x,y)

cmp()函数比较x和y两个对象，并根据比较结果返回一个整数，如果x<y，则返回-1；如果x>y，则返回1,如果x==y则返回0。

```
>>>a=1
>>>b=2
>>>c=2
>>> print cmp(a,b)
-1
>>> print cmp(b,a)
1
>>> print cmp(b,c)
0
```

- divmod(x,y)

divmod(x,y)函数完成除法运算，返回商和余数。

```
>>> divmod(10,3)
(3, 1)
>>> divmod(9,3)
(3, 0)
```

- isinstance(object,class-or-type-or-tuple) -> bool

测试对象类型

```
>>> a='isinstance test'
>>> b=1234
>>> isinstance(a,str)
True
>>> isinstance(a,int)
False
>>> isinstance(b,str)
False
>>> isinstance(b,int)
True
```

- len(object) -> integer

  len()函数返回字符串和序列的长度。

```
>>> len("aa")
2
>>> len([1,2])
2
```

- pow(x,y[,z])

  pow()函数返回以x为底，y为指数的幂。如果给出z值，该函数就计算x的y次幂值被z取模的值。

```
>>> print pow(2,4)
16
>>> print pow(2,4,2)
0
>>> print pow(2.4,3)
13.824
```

- range([lower,]stop[,step])

  range()函数可按参数生成连续的有序整数列表。

```
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(1,10)
[1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(1,10,2)
[1, 3, 5, 7, 9]
```

- round(x[,n])

  round()函数返回浮点数x的四舍五入值，如给出n值，则代表舍入到小数点后的位数。

```
>>> round(3.333)
3.0
>>> round(3)
3.0
>>> round(5.9)
6.0
```

- type(obj)

  type()函数可返回对象的数据类型。

```
>>> type(a)
<type 'list'>
>>> type(copy)
<type 'module'>
>>> type(1)
<type 'int'>
```

- xrange([lower,]stop[,step])

  xrange()函数与range()类似，但xrnage()并不创建列表，而是返回一个xrange对象，它的行为与列表相似，但是只在需要时才计算列表值，当列表很大时，这个特性能为我们节省内存。

```
>>> a=xrange(10)
>>> print a[0]
0
>>> print a[1]
1
>>> print a[2]
2
```

5.2. 内置类型转换函数

- chr(i)

  chr()函数返回ASCII码对应的字符串。

```
>>> print chr(65)
A
>>> print chr(66)
B
>>> print chr(65)+chr(66)
AB
```

- complex(real[,imaginary])

  complex()函数可把字符串或数字转换为复数。

```
>>> complex("2+1j")
(2+1j)
>>> complex("2")
(2+0j)
>>> complex(2,1)
(2+1j)
>>> complex(2L,1)
(2+1j)
```

- float(x)

float()函数把一个数字或字符串转换成浮点数。

```
>>> float("12")
12.0
>>> float(12L)
12.0
>>> float(12.2)
12.199999999999999
```

- hex(x)

hex()函数可把整数转换成十六进制数。

```
>>> hex(16)
'0x10'
>>> hex(123)
'0x7b'
```

- long(x[,base])

long()函数把数字和字符串转换成长整数，base为可选的基数。

```
>>> long("123")
123L
>>> long(11)
11L
```

- list(x)

list()函数可将序列对象转换成列表。如：

```
>>> list("hello world")
['h', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd']
>>> list((1,2,3,4))
[1, 2, 3, 4]
```

- int(x[,base])

int()函数把数字和字符串转换成一个整数，base为可选的基数。

```
>>> int(3.3)
3
>>> int(3L)
3
>>> int("13")
13
>>> int("14",15)
19
```

- min(x[,y,z...])

min()函数返回给定参数的最小值，参数可以为序列。

```
>>> min(1,2,3,4)
1
>>> min((1,2,3),(2,3,4))
(1, 2, 3)
```

- max(x[,y,z...])

  max()函数返回给定参数的最大值，参数可以为序列。

```
>>> max(1,2,3,4)
4
>>> max((1,2,3),(2,3,4))
(2, 3, 4)
```

- oct(x)

  oct()函数可把给出的整数转换成八进制数。

```
>>> oct(8)
'010'
>>> oct(123)
'0173'
```

- ord(x)

  ord()函数返回一个字符串参数的ASCII码或Unicode值。

```
>>> ord("a")
97
>>> ord(u"a")
97
```

- str(obj)

  str()函数把对象转换成可打印字符串。

```
>>> str("4")
'4'
>>> str(4)
'4'
>>> str(3+2j)
'(3+2j)'
```

- tuple(x)

  tuple()函数把序列对象转换成tuple。

```
>>> tuple("hello world")
('h', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd')
>>> tuple([1,2,3,4])
(1, 2, 3, 4)
```

5.3. 序列处理函数

- 常用函数中的len()、max()和min()同样可用于序列。

- filter(function,list)

  调用filter()时，它会把一个函数应用于序列中的每个项，并返回该函数返回真值时的所有项，从而过滤掉返回假值的所有项。

```
>>> def nobad(s):
...     return s.find("bad") == -1
...
>>> s = ["bad","good","bade","we"]
>>> filter(nobad,s)
['good', 'we']
```

  这个例子通过把nobad()函数应用于s序列中所有项，过滤掉所有包含“bad”的项。

- map(function,list[,list])

  map()函数把一个函数应用于序列中所有项，并返回一个列表。

```
>>> import string
>>> s=["python","zope","linux"]
>>> map(string.capitalize,s)
['Python', 'Zope', 'Linux']
```

  map()还可同时应用于多个列表。如：

```
>>> import operator
>>> s=[1,2,3]; t=[3,2,1]
>>> map(operator.mul,s,t)   # s[i]*t[j]
[3, 4, 3]
```

  如果传递一个None值，而不是一个函数，则map()会把每个序列中的相应元素合并起来，并返回该元组。如：

```
>>> a=[1,2];b=[3,4];c=[5,6]
>>> map(None,a,b,c)
[(1, 3, 5), (2, 4, 6)]
```

- reduce(function,seq[,init])

  reduce()函数获得序列中前两个项，并把它传递给提供的函数，获得结果后再取序列中的下一项，连同结果再传递给函数，以此类推，直到处理完所有项为止。

```
>>> import operator
>>> reduce(operator.mul,[2,3,4,5])  # ((2*3)*4)*5
120
>>> reduce(operator.mul,[2,3,4,5],1) # (((1*2)*3)*4)*5
120
>>> reduce(operator.mul,[2,3,4,5],2)  # (((2*2)*3)*4)*5
240
```

- zip(seq[,seq,...])

  zip()函数可把两个或多个序列中的相应项合并在一起，并以元组的格式返回它们，在处理完最短序列中的所有项后就停止。

```
>>> zip([1,2,3],[4,5],[7,8,9])
[(1, 4, 7), (2, 5, 8)]
```

  如果参数是一个序列，则zip()会以一元组的格式返回每个项，如：

```
>>> zip((1,2,3,4,5))
[(1,), (2,), (3,), (4,), (5,)]
>>> zip([1,2,3,4,5])
[(1,), (2,), (3,), (4,), (5,)]
```

Chapter 6. 模块

模块可把一个复杂的程序按功能分开，分别存放到不同文件中，使程序更容易维护和管理。在Python中的模块是一个以.py结尾的Python代码文件。可通过import命令输入，如：

```
import sys
```

import会完成以下三个操作：

- 创建新的名称空间（namespace），该名称空间中拥有输入模块中定义的所有对象；
- 执行模块中的代码；
- 创建该名称空间的变量名。

import语句可同时输入多个模块，如：

```
import os,sys,system
```

也可写成：

```
import os
import sys
import system
```

有些模块的名称很长，我们可在输入时给它起个简单的别名，这样在使用模块中的对象就方便很多，如：

```
import ftplib as ftp
```

有时我们可能只想使用模块中某个对象，又不想把整个模块输入，则可以用from...import语句输入特定对象。如：

```
from ftplib import FTP
```

这样，我们就可直接使用FTP()，而不用带前缀。

如果装载模块出错，会引发ImportError异常。我们可捕获该异常进行相应处理。

Python脚本和模块都是一个以.py结束的文件，那程序是如何判断一个.py文件是作为脚本还是模块呢？关键是一个名为__name__的变量，如果它的值是__main__，则不能作为模块，只能作为脚本直接运行。所以在很多脚本的最后都有一段类似下面的语句，限制只能以脚本方式运行，不作为模块：

```
if __name__ == '__main__':
    main()
```

几个功能相近的模块我们可组成一个Python包，存放到一个目录结构中，通过输入包的路径来调用对象。要定义包，就要建一个与包名同名的目录，接着在该目录下创建__init__.py文件。该文件是包的初始化文件，可以为空，也可定义一个代码。例如一个WebDesign包的目录如下：

```
/WebDesign
        __init_.py
        design.py
        draw.py
        ...
```

我们可通过以下语句输入design模块：

```
import WebDesign.design
```

6.1. String模块

- replace(string,old,new[,maxsplit])

字符串的替换函数，把字符串中的old替换成new。默认是把string中所有的old值替换成new值，如果给出maxsplit值，还可控制替换的个数，如果maxsplit为1，则只替换第一个old值。

```
>>>a="11223344"
>>>print string.replace(a,"1","one")
oneone2223344
>>>print string.replace(a,"1","one",1)
one12223344
```

- capitalize(string)

  该函数可把字符串的首个字符替换成大字。

```
>>> import string
>>> print string.capitalize("python")
Python
```

- split(string,sep=None,maxsplit=-1)

  从string字符串中返回一个列表，以sep的值为分界符。

```
>>> import string
>>> ip="192.168.3.3"
>>> ip_list=string.split(ip,'.')
>>> print ip_list
['192', '168', '3', '3']
```

- join(string[,sep])

  返回用sep连接的字串，默认的sep是空格。

```
>>> import string
>>> a = ['a','b','c']
>>> b = string.join(a,'-')
>>> b
'a-b-c'
>>> a
['a', 'b', 'c']
```

6.2. time模块

内置模块time包含很多与时间相关函数。我们可通过它获得当前的时间和格式化时间输出。

- time()，以浮点形式返回自Linux新世纪以来经过的秒数。在linux中，00:00:00 UTC, January 1, 1970是新纪元的开始。

```
>>> time.time()
1150269086.6630149
>>> time.ctime(1150269086.6630149)
>>> 'Wed Jun 14 15:11:26 2006'
```

- ctime([sec])，把秒数转换成日期格式，如果不带参数，则显示当前的时间。

```
>>> import time
>>> time.ctime()
>>> 'Wed Jun 14 15:02:50 2006'
>>> time.ctime(1138068452427683)
'Sat Dec 14 04:51:44 1901'
```

- sleep(secs)，定时。

```
>>> time.sleep(10)
>>>                #10秒后才会出现>>>提示符
```

Chapter 7. 类

类是面向对象编程的一个重要概念。通过类的创建和继承，可重用代码，减少代码复杂度。Python是一种面向对象的脚本语言，用class语句可创建类，语法规则如下：

```
class classnmae([class_parent,...]):
      ...
      def method():
          ...
...
```

一个例子：

```
#!/usr/bin/python
#-*- encoding:utf-8 -*-

class test:                                 #定义一个test类
        desc = "这是一个测试类。"            #在类中定义一个属性desc

        def __init__(self,name1):           #对象构造函数，初始化类
                self.name1 = name1

        def show(self,name2):               #在类中定义一个方法show()
                print "hello world"
                print 'name1:',self.name1
                print 'name2:',name2

instance = test('这是传递给name1的值')    #生成test类的实例对象instance

print instance.desc                         #调用类中的desc属性

instance.show('这是传递给name2的值')      #调用类中的show()方法
```

把该脚本命名为test.py，并用chmod +x test.py使脚本有执行的权限 ，运行该脚本结果如下：

```
debian:~/python# ./test.py
这是一个测试类。
hello world
name1: 这是传递给name1的值
name2: 这是传递给name2的值
```

这里只是Python语言中类的一个简单介绍。详细介绍可参考网站上自由文档栏目中的Python资料。

Chapter 8. 异常处理

Python的异常处理能力是很强大的，可向用户准确反馈出错信息。在Python中，异常也是对象，可对它进行操作。所有异常都是基类Exception的成员。异常处理的try语法有两种，一种是：

```
try:
   block
except [exception,[data...]]:
   block
else:
   block
```

该种异常处理语法的规则是：

- 执行try下的语句，如果引发异常，则执行过程会跳到第一个except语句。
- 如果第一个except中定义的异常与引发的异常匹配，则执行该except中的语句。
- 如果引发的异常不匹配第一个except，则会搜索第二个except，允许编写的except数量没有限制。
- 如果所有的except都不匹配，则异常会传递到下一个调用本代码的最高层try代码中。
- 如果没有发生异常，则执行else块代码。

try语句的第二种语法是：

```
try:
   block
finally:
   block
```

该语句的执行规则是：

- 执行try下的代码。
- 如果发生异常，在该异常传递到下一级try时，执行finally中的代码。
- 如果没有发生异常，则执行finally中的代码。

第二种try语法在无论有没有发生异常都要执行代码的情况下是很有用的。例如我们在python中打开一个文件进行读写操作，我在操作过程中不管是否出现异常，最终我都是要把该文件关闭的。

除了系统引发的异常外，我们还可用raise语句手工引发一个异常：

```
raise [exception[,data]]
```

Chapter 9. 文件处理

文件是我们储存信息的地方，我们经常要对文件进行读、写、删除等的操作，在Python中，我们可用Python提供的函数和方法方便地操作文件。

9.1. 文件处理的函数和方法

使用Open()函数可打开文件，语法格式如下：

```
file_handler = open(filename,[,mode[,bufsize]]
```

filename是你要操作的文件名，如果不在当前路径，需指出具体路径。mode是打开文件的模式，表示你要如何操作文件，bufsize表示是否使用缓存。

Table 9.1. mode

模式	描述
r	以读方式打开文件，可读取文件信息。
w	以写方式打开文件，可向文件写入信息。
a	以追加方式打开文件，文件指针自动移到文件尾。
r+	以读写方式打开文件，可对文件进行读和写操作。
w+	消除文件内容，然后以读写方式打开文件。
a+	以读写方式打开文件，并把文件指针移到文件尾。
b	以二进制模式打开文件，而不是以文本模式。该模式只对Windows或Dos有效，类Unix的文件是用二进制模式进行操作的。

Table 9.2. bufsize

bufsize取值	描述
0	禁用缓冲
1	行缓冲
>1	指定缓冲区的大小
<1	系统默认的缓冲区大小

open()函数返回一个文件对象，我们可通过read()或write()函数对文件进行读写操作，下面是一些文件对象方法：

Table 9.3. 文件对象方法

方法	描述
f.close()	关闭文件，记住用open()打开文件后一定要记得关闭它，否则会占用系统的可打开文件句柄数。
f.fileno()	获得文件描述符
f.flush()	刷新输出缓存
f.isatty()	如果文件是一个交互终端，则返回True，否则返回False。
f.read([count])	读出文件，如果有count，则读出count个字节。
f.readline()	读出一行信息。
f.readlines()	读出所有行，也就是读出整个文件的信息。
f.seek(offset[,where])	把文件指针移动到相对于where的offset位置。offset为0表示文件开始处，这是默认值 ；1表示当前位置；2表示文件结尾。
f.tell()	获得文件指针位置。
f.truncate([size])	截取文件，使文件的大小为size。
f.write(string)	把string字符串写入文件。
f.writelines(list)	把list中的字符串一行一行地写入文件。

9.2. 示例

- 文件的打开或创建

```
#!/usr/bin/env python
#-*- encoding:UTF-8 -*-

filehandler = open('test.txt','w')                #以写模式打开文件，如果文件不存在则创建
filehandler.write('this is a file open/create test.\nthe second line.')

filehandler.close()
```

```
#!/usr/bin/env python
#-*- encoding:UTF-8 -*-

filehandler = open('test.txt','a')      #以追加模式打开文件，如果文件不存在则创建

filehandler.write('\nappend the text in another line.\n')

filehandler.close()
```

- 读取文件

```
#!/usr/bin/env python
#-*- encoding:UTF-8 -*-

filehandler = open('test.txt','r')    #以读方式打开文件，rb为二进制方式(如图片或可执行文件等)

print 'read() function:'              #读取整个文件
print filehandler.read()

print 'readline() function:'          #返回文件头，读取一行
filehandler.seek(0)
print filehandler.readline()

print 'readlines() function:'         #返回文件头，返回所有行的列表
filehandler.seek(0)
print filehandler.readlines()

print 'list all lines'                #返回文件头，显示所有行
filehandler.seek(0)
textlist = filehandler.readlines()
for line in textlist:
      print line

print 'seek() function'               #移位到第32个字符，从33个字符开始显示余下内容
filehandler.seek(32)
print filehandler.read()

print 'tell() function'               #移位到文件头，从头开始显示2位字符
filehandler.seek(0)
print filehandler.readline()          #显示第一行内容
print filehandler.tell()              #显示当前位置
print filehandler.readline()          #显示第二行内容
print filehandler.read()              #显示余下所有内容

filehandler.close()                   #关闭文件句柄
```

- 文件系统操作

```
#!/usr/bin/env python
#-*- encoding:utf-8 -*-

import os,fnmatch,glob

for fileName in os.listdir ( '/root' ):                 #列出/root目录内容，不包括.和..
   print fileName

os.mkdir('py')                  #在当前目录下创建一个py目录，且只能创建一层
os.rmdir( 'py')                 #在当前目录下删除py目录，且只能删除一层
os.makedirs('py/aa')            #可创建多层目录
os.removedirs('py/aa')          #可删除多层目录

print 'demonstration fnmatch module'
for fileName in os.listdir ( '/root/python/file' ):
        if fnmatch.fnmatch(fileName,'*.txt'):         #利用UNIX风格的通配，只显示后缀为txt
                print fileName

print 'demonstration glob module'
for fileName in glob.glob ( '*.txt' ):                #利用UNIX风格的通配，只显示后缀为txt
        print fileName
```

- 获取文件状态

```
#!/usr/bin/env python
#-*- encoding:UTF-8 -*-

import os,time,stat

fileStats = os.stat ( 'test.txt' )                          #获取文件/目录的状态
fileInfo = {
'Size':fileStats [ stat.ST_SIZE ],                          #获取文件大小
'LastModified':time.ctime( fileStats [ stat.ST_MTIME ] ),   #获取文件最后修改时间
'LastAccessed':time.ctime( fileStats [ stat.ST_ATIME ] ),   #获取文件最后访问时间
'CreationTime':time.ctime( fileStats [ stat.ST_CTIME ] ),   #获取文件创建时间
'Mode':fileStats [ stat.ST_MODE ]                           #获取文件的模式
}
#print fileInfo

for field in fileInfo:                                      #显示对象内容
        print '%s:%s' % (field,fileInfo[field])

#for infoField,infoValue in fileInfo:
#       print '%s:%s' % (infoField,infoValue)
if stat.S_ISDIR ( fileStats [ stat.ST_MODE ] ):              #判断是否路径
        print 'Directory. '
else:
        print 'Non-directory.'

if stat.S_ISREG ( fileStats [ stat.ST_MODE ] ):              #判断是否一般文件
   print 'Regular file.'
elif stat.S_ISLNK ( fileStats [ stat.ST_MODe ] ):            #判断是否链接文件
   print 'Shortcut.'
elif stat.S_ISSOCK ( fileStats [ stat.ST_MODe ] ):           #判断是否套接字文件
   print 'Socket.'
elif stat.S_ISFIFO ( fileStats [ stat.ST_MODe ] ):           #判断是否命名管道
   print 'Named pipe.'
elif stat.S_ISBLK ( fileStats [ stat.ST_MODe ] ):            #判断是否块设备
   print 'Block special device.'
elif stat.S_ISCHR ( fileStats [ stat.ST_MODe ] ):            #判断是否字符设置
   print 'Character special device.'
```

```
#!/usr/bin/env python
#-*- encoding:UTF-8 -*-

import os.path

fileStats = 'test.txt'

if os.path.isdir ( fileStats ):         #判断是否路径
        print 'Directory.'
elif os.path.isfile ( fileStats ):      #判断是否一般文件
        print 'File.'
elif os.path.islink ( fileStats ):      #判断是否链接文件
        print 'Shortcut.'
elif os.path.ismount ( fileStats ):     #判断是否挂接点
        print 'Mount point.'
```

stat模块描述了os.stat(filename)返回的文件属性列表中各值的意义。我们可方便地根据stat模块存取os.stat()中的值。

- 串行化文件

```
#!/usr/bin/env python
#-*- encoding:UTF-8 -*-

import pickle

filehandler = open('pickle.txt','w')

text = ['this is a pickle demonstrate','aa','bb']

pickle.dump(text,filehandler)           #把text的内容序列化后保存到pickle.txt文件中

filehandler.close()

filehandler2 = open('pickle.txt')

textlist = pickle.load(filehandler2)    #还原序列化字符串
print textlist

filehandler2.close()

#cpickle是用C写的pickle模块，比标准的pickle速度快很多，使用方法同pickle。
```

- 内存文件

```
#!/usr/bin/env python
#-*- coding: utf-8 -*-

import StringIO

fileHandle = StringIO.StringIO ( "Let freedom ring." )   #create file in memory

print fileHandle.read() # "Let freedom ring."

fileHandle.close()

#cStringIO是用C写的StringIO模块，执行速度比StringIO快。
```

shutil模块是一个高级的文件处理模块，可实现文件的拷贝、删除等操作。

Chapter 10. 正则表达式

正则表达式是一个很有用的工具，可处理复杂的字符匹配和替换工作。在Python中内置了一个re模块以支持正则表达式。

正则表达式有两种基本的操作，分别是匹配和替换。

- 匹配就是在一个文本字符串中搜索匹配一特殊表达式；
- 替换就是在一个字符串中查找并替换匹配一特殊表达式的字符串。

10.1. 基本元素

正则表达式定义了一系列的特殊字符元素以执行匹配动作。

Table 10.1. 正则表达式基本字符

字符	描述
text	匹配text字符串
.	匹配除换行符之外的任意一个单个字符
^	匹配一个字符串的开头
$	匹配一个字符串的末尾

在正则表达式中，我们还可用匹配限定符来约束匹配的次数。

Table 10.2. 匹配限定符

最大匹配	最小匹配	描述
*	*	重复匹配前表达式零次或多次
+	+	重复匹配前表达式一次或多次
?	?	重复匹配前表达式零次或一次
{m}	{m}	精确重复匹配前表达式m次
{m,}	{m,}	至少重复匹配前表达式m次
{m,n}	{m,n}	至少重复匹配前表达式m次，至多重复匹配前表达式n次

据上所述，".*"为最大匹配，能匹配源字符串所有能匹配的字符串。".* "为最小匹配，只匹配第一次出现的字符串。如：d.*g能匹配任意以d开头，以g结尾的字符串，如"debug"和"debugging"，甚至"dog is walking"。而d.* g只能匹配"debug"，在"dog is walking"字符串中，则只匹配到"dog "。

在一些更复杂的匹配中，我们可用到组和运算符。

Table 10.3. 组和运算符

<table>
<tr><th>组</th><th>描述</th></tr>
<tr><td>[...]</td><td>匹配集合内的字符，如[a-z],[1-9]或[,./;']</td></tr>
<tr><td>...</td><td>匹配除集合外的所有字符，相当于取反操作</td></tr>
<tr><td>A|B</td><td>匹配表达式A或B，相当于OR操作</td></tr>
<tr><td>(...)</td><td>表达式分组，每对括号为一组，如([a-b]+)([A-Z]+)([1-9]+)</td></tr>
<tr><td>\number</td><td>匹配在number表达式组内的文本</td></tr>
</table>

有一组特殊的字符序列，用来匹配具体的字符类型或字符环境。如\b匹配字符边界，food\b匹配"food"、"zoofood"，而和"foodies"不匹配。

Table 10.4. 特殊字符序列

字符	描述
\A	只匹配字符串的开始
\b	匹配一个单词边界
\B	匹配一个单词的非边界
\d	匹配任意十进制数字字符，等价于r'[0-9]'
\D	匹配任意非十进制数字字符，等价于r'0-9'
\s	匹配任意空格字符（空格符、tab制表符、换行符、回车、换页符、垂直线符号）
\S	匹配任意非空格字符
\w	匹配任意字母数字字符
\W	匹配任意非字母数字字符
\Z	仅匹配字符串的尾部
\	匹配反斜线字符

有一套声明(assertion)对具体事件进行声明。

Table 10.5. 正则表达式声明

声明	描述
(iLmsux)	匹配空字符串，iLmsux字符对应下表的正则表达式修饰符。
(:...)	匹配圆括号内定义的表达式，但不填充字符组表。
(P<name>)	匹配圆括号内定义的表达式，但匹配的表达式还可用作name标识的符号组。
(P=name)	匹配所有与前面命名的字符组相匹配的文本。
(#...)	引入注释，忽略圆括号内的内容。
(=...)	如果所提供的文本与下一个正则表达式元素匹配，这之间没有多余的文本就匹配。这允许在一个表达式中进行超前操作，而不影响正则表达式其余部分的分析。如"Martin"其后紧跟"Brown"，则"Martin(=Brown)"就只与"Martin"匹配。
(!...)	仅当指定表达式与下一个正则表达式元素不匹配时匹配，是(=...)的反操作。
(<=...)	如果字符串当前位置的前缀字符串是给定文本，就匹配，整个表达式就在当前位置终止。如(<=abc)def表达式与"abcdef"匹配。这种匹配是对前缀字符数量的精确匹配。
(<!...)	如果字符串当前位置的前缀字符串不是给定的正文，就匹配，是(<=...)的反操作。

正则表达式还支持一些处理标志，它会影响正则式的执行方法。

Table 10.6. 处理标志

标志	描述
I或IGNORECASE	忽略表达式的大小写来匹配文本。

10.2. 操作

通过re模块，我们就可在python中利用正则式对字符串进行搜索、抽取和替换操作。如：re.search()函数能执行一个基本的搜索操作，它能返回一个MatchObject对象。re.findall()函数能返回匹配列表。

```
>>> import re
>>> a="this is my re module test"
>>> obj = re.search(r'.*is',a)
>>> print obj
<_sre.SRE_Match object at 0xb7d7a218>
>>> obj.group()
'this is'
>>> re.findall(r'.*is',a)
['this is']
```

MatchObject对象方法

Table 10.7. MatchObject对象方法

方法	描述
expand(template)	展开模板中用反斜线定义的内容。
m.group([group,...])	返回匹配的文本，是个元组。此文本是与给定group或由其索引数字定义的组匹配的文本，如果没有组定组名，则返回所有匹配项。
m.groups([default])	返回一个元组，该元组包含模式中与所有组匹配的文本。如果给出default参数，default参数值就是与给定表达式不匹配的组的返回值。default参数的默认取值为None。
m.groupdict([default])	返回一个字典，该字典包含匹配的所有子组。如果给出default参数，其值就是那些不匹配组的返回值。default参数的默认取值为None。
m.start([group])	返回指定group的开始位置，或返回全部匹配的开始位置。
m.end([group])	返回指定group的结束位置，或返回全部匹配的结束位置。
m.span([group])	返回两元素组，此元组等价于关于一给定组或一个完整匹配表达式的(m.start(group),m.end(group)))列表
m.pos	传递给match()或search()函数的pos值。
m.endpos	传递给match()或search()函数的endpos值。
m.lastindex	
m.lastgroup	
m.re	创建这个MatchObject对象的正则式对象
m.string	提供给match()或search()函数的字符串。

使用sub()或subn()函数可在字符串上执行替换操作。sub()函数的基本格式如下：

```
sub(pattern,replace,string[,count])
```

示例

```
>>> str = 'The dog on my bed'
>>> rep = re.sub('dog','cat',str)
>>> print rep
The cat on my bed
```

replace参数可接受函数。要获得替换的次数，可使用subn()函数。subn()函数返回一个元组，此元组包含替换了的文本和替换的次数。

如果需用同一个正则式进行多次匹配操作，我们可把正则式编译成内部语言，提高处理速度。编译正则式用compile()函数来实现。compile()函数的基本格式如下：

```
compile(str[,flags])
```

str表示需编译的正则式串，flags是修饰标志符。正则式被编译后生成一个对象，该对象有多种方法和属性。

Table 10.8. 正则式对象方法/属性

方法/属性	描述
r.search(string[,pos[,endpos]])	同search()函数，但此函数允许指定搜索的起点和终点
r.match(string[,pos[,endpos]])	同match()函数，但此函数允许指定搜索的起点和终点
r.split(string[,max])	同split()函数
r.findall(string)	同findall()函数
r.sub(replace,string[,count])	同sub()函数
r.subn(replace,string[,count])	同subn()函数
r.flags	创建对象时定义的标志
r.groupindex	将r'(Pid)'定义的符号组名字映射为组序号的字典
r.pattern	在创建对象时使用的模式

转义字符串用re.escape()函数。

通过getattr获取对象引用

```
>>> li=['a','b']
>>> getattr(li,'append')
>>> getattr(li,'append')('c')          #相当于li.append('c')
>>> li
['a', 'b', 'c']
>>> handler=getattr(li,'append',None)
>>> handler
<built-in method append of list object at 0xb7d4a52c>
>>> handler('cc')                      #相当于li.append('cc')
>>> li
['a','b','c','cc']
>>>result = handler('bb')
>>>li
['a','b','c','cc','bb']
>>>print result
None
```

Chapter 11. 调试

Python自带了一个调试器叫pdb，和Gnu的gbd类似。下面用一个简单的程序来演示pdb的功能。程序代码如下：

```
#!/usr/bin/python

import pdb
a = "aaa"
pdb.set_trace()
b = "bbb"
c = "ccc"
final = a + b + c
print final
```

该程序已导入pdb模块，并在代码中添加的pdb.set_trace()跟踪点。现在让我们来运行该程序。

```
localhost:~/python/pdb# python pdbtest.py
--Return--
> /usr/lib/python2.3/pdb.py(992)set_trace()->None
-> Pdb().set_trace()              # 从跟踪点开始执行
(Pdb) n                           # n 读入下一行代码
> /root/python/pdb/pdbtest.py(6) ()
-> b = "bbb"
(Pdb) n
> /root/python/pdb/pdbtest.py(7) ()
-> c = "ccc"
(Pdb) p b                         # p 打印变量值
'bbb'
(Pdb) l                           # l 显示当前执行位置
  2
  3     import pdb
  4     a = "aaa"
  5     pdb.set_trace()
  6     b = "bbb"
  7  -> c = "ccc"
  8     final = a + b + c
  9     print final
 10
[EOF]
(Pdb) n
> /root/python/pdb/pdbtest.py(8) ()
-> final = a + b + c
(Pdb) n                           # 如果命令和上次的一样，也可直接按回车，不用输入'n'
> /root/python/pdb/pdbtest.py(9) ()
-> print final
(Pdb) n
aaabbbccc
--Return--
> /root/python/pdb/pdbtest.py(9) ()->None
-> print final
(Pdb) p a,b,c,final
('aaa', 'bbb', 'ccc', 'aaabbbccc')
(Pdb)
('aaa', 'bbb', 'ccc', 'aaabbbccc')
(Pdb) n
localhost:~/python/pdb#           # 返回shell
```

pdb还有很多命令，用help命令就可以列出所有的pdb命令，用help p可以查询p命令的说明。

Chapter 12. HOW-TO

本章内容记录Python的一些小技巧小知识。来源是网上摘录或自己学习所得。

- 如何判断操作系统类型

```
import sys
print sys.platform
print sys.version
```

- 显示和修改python的Module搜索路径

```
>>> import sys
>>> print sys.path
['', '/usr/lib/python23.zip', '/usr/lib/python2.3', '/usr/lib/python2.3/plat-linux2',
 '/usr/lib/python2.3/lib-tk', '/usr/lib/python2.3/lib-dynload', '/usr/local/lib/pytho
 '/usr/lib/python2.3/site-packages']
>>> sys.path.append('/usr/lib/mypath')
>>> print sys.path
['', '/usr/lib/python23.zip', '/usr/lib/python2.3', '/usr/lib/python2.3/plat-linux2',
'/usr/lib/python2.3/lib-tk', '/usr/lib/python2.3/lib-dynload', '/usr/local/lib/python
 '/usr/lib/python2.3/site-packages', '/usr/lib/mypath']
```

- 把列表转换成字符串

```
>>> t=['a','b','c']
>>> print t
['a', 'b', 'c']
>>> import string
>>> print string.join(t)
a b c
```

- 运行系统程序

```
>>>import os
>>>os.system('ls')            #用os.system()可执行系统命令
>>>exec "os.system('ls')"     #用exec可执行字符串中的命令，两个命令的效果一样。
```

以上两个命令的输出都是直接显示在屏幕上，不能保存到变量中，如果我们要把输出保存起来，可用os.pope\ n()函数。

```
>>>cmd = '/usr/bin/mkntpwd %s' % password
>>>handler = os.popen(cmd,'r')
>>>passwordString=handler.read()   #passwordString为mkntpwd程序的输出结果
```

使用commands模块也可以获取程序的输出，它包含一些基于os.popen()的封装函数，使我们能更方便地获取运行系统命令和获取命令的输出，但该模块只在Unix系统下有效，不能用于Windows平台。

```
>>> import commands
>>> status,output = commands.getstatusoutput('ls -l')
>>> print output
总计 96564
-rw-r--r--  1 root     root         4459 2005-12-01 10:23 2005.sxw
-rw-r--r--  1 root     root        27511 2006-04-12 16:54 20060412_user.ods
-rw-r--r--  1 root     root       202258 2006-01-06 16:48 2006风景-1月.jpg
...
>>> print status
0
```

在Python2.4中引入一个新的模块叫subprocess，用于取代os.system、os.spawn*、os.popen*、popen2.*、commands.*。

- 编码转换

```
#!/usr/bin/python
#-*-coding:utf-8 -*-

a=u"测试"
b=a.encode('gb2312')
print a
print b
```

- 交换两个变量

```
>>> a,b = 1,2
>>> a,b
(1, 2)
>>> a,b = b,a
>>> a,b
(2, 1)
>>> a
2
>>> b
1
```

- 测试数据类型

```
>>> a=123
>>> b='test'
>>> a
123
>>> b
'test'
>>> isinstance(a,int)
True
>>> isinstance(a,str)
False
>>> isinstance(b,int)
False
>>> isinstance(b,str)
True
```

- 用in判断是否包含子字符串

```
>>> a='this is my test'
>>> 'is' in a
True
>>> 'mm' in a
False
```

- __iter__迭代器

```
>>> a = "iterator"
>>> t = iter(a)
>>> t.next()
'i'
>>> t.next()
't'
>>> t.next()
'e'
>>> t.next()
'r'
>>> t.next()
'a'
>>> t.next()
't'
>>> t.next()
'o'
>>> t.next()
'r'
>>> t.next()
Traceback (most recent call last):
  File "<stdin>", line 1, in
StopIteration
```

自已写一个迭代器类

```
>>> class reverse:
...     def __init__(self,data):
...             self.data=data
...             self.index=len(data)
...     def __iter__(self):
...             return self
...     def next(self):
...             if self.index == 0:
...                     raise StopIteration
...             self.index = self.index - 1
...             return self.data[self.index]
...
>>> for char in reverse('iterator'):
...     print char
...
r
o
t
a
r
e
t
i
>>>
```

- 通过getattr可以得到一个在运行时才知道具体函数名的对象的引用，能增强我们程序的灵活性。

```
>>> li=['a','b']
>>> getattr(li,'append')
>>> getattr(li,'append')('c')          #相当于li.append('c')
>>> li
['a', 'b', 'c']
>>> handler=getattr(li,'append',None)
>>> handler
<built-in method append of list object at 0xb7d4a52c>
>>> handler('cc')                      #相当于li.append('cc')
>>> li
['a','b','c','cc']
>>>result = handler('bb')
>>>li
['a','b','c','cc','bb']
>>>print result
None
```

编程示例：

```
import statsout

def output(data, format="text"):
    output_function = getattr(statsout, "output_%s" % format)
    return output_function(data)
```

以上代码表示，output函数接收一个data参数和format参数，根据format参数的值，从statsout模块中取出output_text函数运行，data参数通过output_function(data)传递给了statsout模块中的output_text函数。format取不同值可从statsout模块中取出不同的函数运行（output_xxxx）。也就是说我们要运行的函数是在程序运行后才确定的。这样我们可把不同的函数以output_xxx形式命名放在statout模块中，通过以上程序可动态调用各种函数。

- hasattr用于确定一个对象是否具有某个属性。

  语法：

```
hasattr(object, name) -> bool
```

  判断object中是否有name属性，返回一个布尔值。

- 拆分序列

```
>>> a=[c for c in 'abcdefg']
>>> a
['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>>
```

  按if条件拆分序列

```
>>> a=[c for c in '123456' if int(c)<3]      如果if的条件为真，则执行for循环
>>> a
['1', '2']
>>> a=[c for c in '123456' if int(c)>3]      如果if的条件为假，则不执行for循环
>>> a
['4', '5', '6']
```

- __dict__记录模块或类中所有对象的信息，它以字典{name:object}的形式记录这些信息，如果wikiaction是一个模块，则可以这样显示：

```
>>>import wikiaction
>>>print wikiaction.__dict__
{'do_test': <function do_test at 0xb7c10534>, 'do_diff': <function do_diff at 0xb7c0e
ction do_refresh at 0xb7c1025c>, 'do_userform': <function do_userform at 0xb7c103e4>,
 getHandler at 0xb7c105a4>, 'do_raw': <function do_raw at 0xb7c10454>, 'do_chart': <f
c104c4>, 're': <module 're' from '/usr/lib/python2.3/re.pyc'>, 'pysupport': <module '
 from '/usr/lib/python2.3/site-packages/MoinMoin/util/pysupport.pyc'>, 'config': <moc
om '/usr/lib/python2.3/site-packages/MoinMoin/config.pyc'>}
```

- 'and'的特殊用法

```
>>> 'a' and 'b'         #如果两个都为真值，返回最后一个真值
'b'
>>> 'b' and 'a'         #同上
'a'
>>> 'a' and 'b' and 'c' #同上
'c'
>>> '' and 'a'          #如果有假值，则返回假值
''
>>> 'a' and '' and 'c'  #同上
''
>>> '' and 0            #如果两个都为假值，返回第一个假值
''
>>> 0 and ''            #同上
0
```

- 'or'的的特殊用法

```
>>> 'a' or 'b'          #如果有一个为真值，则返回第一个真值
'a'
>>> 'b' or 'a'          #同上
'b'
>>> 'a' or 'b' or ''    #同上
'a'
>>> 0 and '' and {}     #如果所有都是假值，则返回第一个假值
0
>>> {} and '' and {}    #同上
{}
```

- lambda匿名函数的用法

```
>>> a=lambda c:c*2
>>> a
<function <lambda> at 0xb7dd710c>
>>> a(2)
4
>>> a(5)
10
```

Python 学习笔记 模块篇

整理：Jims of 肥肥世家

jims.yang@gmail.com

发布时间：2004年7月10日

更新时间：2006年03月01日，增加cjkcodecs模块。

Abstract

Python为开发人员提供了丰富的模块，通过这些模块，我们就可快速开发出功能强大的程序。本笔记记录我所接触或学习过的Python模块，为想学习Python的朋友提供一个参考。

Table of Contents

Chapter 1. Python Imaging Library(PIL)

PIL(Python图形库)为python提供强大的图形处理的能力，并提供广泛的图形文件格式支持，当前最新的版本是1.1.4。可到以下网址http://www.pythonware.com/products/pil/index.htm了解PIL的最新动态。该库能进行图形格式的转换、打印和显示。还能进行一些图形效果的处理，如图形的放大、缩小和旋转等。是Python用户进行图象处理的强有力工具。

1.1. 安装

1.1.1. 下载相关软件

- 到http://www.pythonware.com/products/pil/index.htm下载最新版的PIL安装程序。这里介绍的是在linux下的安装方法。windows平台的安装方法较简单，只要双击安装程序，就可一步步安装好了。

- 如果要PIL支持jpeg格式文件，还需安装jpeg库文件，可到http://www.ijg.org下载，现时最新的版本是jpegsrc.v6b.tar.gz。

- 如果要PIL支持压缩功能，还要下载Zlib库，可到http://www.gzip.org/zlib/下载zlib-1.1.4.tar.gz。

1.1.2. 开始安装

- 先安装jpeg库，输入以下命令进行安装：

```
tar xfz jpegsrc.v6b.tar.gz
cd jpeg-6b
./configure
make
make test
make install
make install-lib
```

- 接着安装Zlib库，输入以下命令进行安装：

```
tar xfz zlib-1.1.4.tar.gz
cd zlib-1.1.4
./configure
make
make install
```

- 最后安装PIL，输入以下命令进行安装：

```
tar xfz Imaging-1.1.4.tar.gz
cd Imaging-1.1.4
cd libImaging
./configure
make
cd ..
python setup.py build
python setup.py install
```

- 测试安装是否成功，可以在Python的命令行界面输入以下代码：

```
>>>import Image
>>>im = Image.open("test.jpg")
>>>im.show()
```

如果成功打开test.jpg图片则安装成功。注意，在linux中，需要用xv程序来显示图片，所以如果没装xv，python会提示找不到xv。可到http://www.trilon.com/xv/downloads.html下载xv。

Chapter 2. Pmw(Python megawidgets)Python超级GUI组件集

Pmw是一个在python中利用Tkinter模块构建的高级GUI组件，每个Pmw都合并了一个或多个Tkinter组件，以实现更有用和更复杂的功能。如，Pmw中的一个ScrolledListBox（滚动列表框）实现了Tkinter的Scrollbar（滚动条）和ListBox（列表框）功能，使我们编程更方便。如果你在Python中开发GUI程序，Pmw是将是你的一个好帮手。

2.1. 安装

现在最新的Pmw是1.2版，Pmw的安装比较简单，只要到http://pmw.sourceforge.net/下载软件，然后用tar -zxvf命令解压文件，把解压出来的Pmw目录拷到python的模块目录下就可以了，如site-packages 目录。windows平台使用同一压缩包，安装方法也一样。安装完成后可登录进python的命令行界面运行“import Pmw”测试是否安装成功，如果没有出错信息，则安装成功，可以使用了。

2.2. 模块功能演示

2.2.1. ScrolledListBox(滚动列表框)

```
#ScrolledListBox used to select image.

from Tkinter import *
import Pmw

class ImageSelection( Frame ):
    """List of available images and an area to display them"""

    def __init__( self, images ):
        """Create list of PhotoImages and Label to display them"""

        Frame.__init__( self )
        Pmw.initialise()
        self.pack( expand = YES, fill = BOTH )
        self.master.title( "Select an image" )

        self.photos = []

        #add PhotoImage object to list photos
        for item in images:
            self.photos.append( PhotoImage( file = item ) )

        #create scrolled list box with vertical scrollbar
        self.listBox = Pmw.ScrolledListBox( self, items = images,
                                            listbox_height = 3,
                                            vscrollmode = "static",
                                            selectioncommand = self.switchImage )
        self.listBox.pack( side = LEFT, expand = YES, fill = BOTH, padx = 5, pady = 5 )

        self.display = Label( self, image = self.photos[0] )
        self.display.pack( padx = 5, pady = 5 )

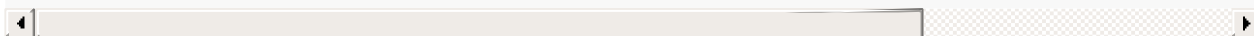
    def switchImage ( self ):
        """Change image in Label to current selection"""

        #get tuple containing index of selected list item
        chosenPicture = self.listBox.curselection()

        #configure label to display selected image
        if chosenPicture:
            choice = int( chosenPicture[0] )
            self.display.config( image = self.photos[ choice ] )

def main():
    images = [ "c:\python23\logo.gif", "c:\python23\china.gif", "c:\python23\canada.gif",
    ImageSelection(images).mainloop()

if __name__ == "__main__":
    main()
```





2.2.2. ScrolledText（滚动文本框）

```
#Copying selected text from one text area to another.

from Tkinter import *
import Pmw

class CopyTextWindow( Frame ):
    """Demonatrate ScrolledText"""

    def __init__( self ):
        """Create two ScrolledText and a Button"""

        Frame.__init__( self )
        Pmw.initialise()
        self.pack( expand = YES, fill = BOTH )
        self.master.title( "ScrolledText Demo" )

        #create scrolled text box with word wrap enable
        self.text1 = Pmw.ScrolledText( self, text_width = 25, text_height = 12,
                                       text_wrap = WORD, hscrollmode = "static",
                                       vscrollmode = "static" )
        self.text1.pack( side = LEFT, expand = YES, fill = BOTH, padx = 5, pady = 5 )

        self.copyButton = Button( self, text = "Copy >>>", command = self.copyText )
        self.copyButton.pack( side = LEFT, padx = 5, pady = 5 )

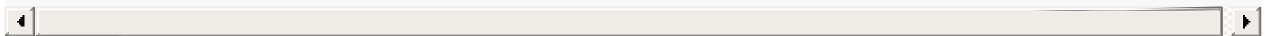
        #create uneditable scrolled text box
        self.text2 = Pmw.ScrolledText( self, text_state = DISABLED, text_width = 25,
                                       text_height = 12, text_wrap = WORD,
                                       hscrollmode = "static", vscrollmode = "static" )
        self.text2.pack( side = LEFT, expand = YES, fill = BOTH, padx = 5, pady = 5 )

    def copyText( self ):
        """set the text in the second ScrolledText"""

        self.text2.settext( self.text1.get( SEL_FIRST,SEL_LAST ) )

def main():
    CopyTextWindow().mainloop()

if __name__ == "__main__":
    main()
```





Chapter 3. PyXML

PyXML是一套用Python解析和处理XML文档的工具包，包中的4DOM是完全相容于W3C DOM规范的。它包含以下内容：

- xmlproc: 一个符合规范的XML解析器。
- Expat: 一个快速的，非验证的XML解析器。
- sgmlop: a C helper module that can speed-up xmllib.py and sgmllib.py by a factor of 5.
- PySAX: SAX 1 and SAX2 libraries with drivers for most of the parsers.
- 4DOM: A fully compliant DOM Level 2 implementation

- javadom: An adapter from Java DOM implementations to the standard Python DOM binding.
- pulldom: a DOM implementation that supports lazy instantiation of nodes.
- marshal: a module with several options for serializing Python objects to XML, including WDDX and XML-RPC.

3.1. 安装

到http://sourceforge.net/project/showfiles.php group_id=6473下载最新版的模块，现在是PyXML-0.8.3。安装PyXML需要有python2.0以上及以上的版本。下载完成后用tar解压缩生成PyXML-0.8.3目录，进入该目录并运行python setup.py build和python setup.py install完成安装。测试方法是进入命令行交互界面运行“import xml.dom.ext"命令，如果没提示模块出错则说明安装成功。PyXML提供windows平台的安装包，下载后双击运行就可以了。

3.2. 使用

由于该模块的内容较多，所以该模块的详细使用将我在“PyXML学习笔记”中单独讨论。

Chapter 4. PyGame

PyGame是一组用于多媒体开发和游戏软件开发的模块。

Chapter 5. PyOpenGL

PyOpenGL模块封装了“OpenGL应用程序编程接口”，通过该模块python程序员可在程序中集成2D和3D的图形。

Chapter 6. NumPy和Numarray

NumPy是Python的一个扩展库，主要用于处理任意维数的固定类型数组，它的低层代码使用C来编写，所以速度的优势很明显。Numarray是NumPy的一个改进版，用于取代NumPy。

Chapter 7. MySQLdb

MySQLdb模块用于连接MySQL数据库。源码位于http://sourceforge.net/projects/mysql-python，这里还有用于zope的ZMySQLDA模块，通过它就可在zope中连接mysql数据库。

7.1. 安装

安装的方法在解压目录的README文件中有详细说明。不难，这里就不详细讲了。要注意的一点是，如果你的mysql不是安装在默认的路径，而是安装在/usr/local/mysql这样的路径的话，libmysqlclient.so.12这个动态库python可能会找不到，造成import出错，解决方法是在/usr/lib下做一个符号连接，ln -s /usr/local/mysql/lib/mysql/libmysqlclient.so.12 libmysqlclient.so.12。最后在python中用import MySQLdb测试，如果没有出错信息就说明安装成功，可以连接mysql数据库了。

7.2. 模块功能

- connect()方法用于连接数据库，返回一个数据库连接对象。如果要连接一个位于host.remote.com服务器上名为fourm的MySQL数据库，连接串可以这样写：

  ```
  db = MySQLdb.connect(host="remote.com",user="user",passwd="xxx",db="fourm" )
  ```

  connect()的参数列表如下：

  - host，连接的数据库服务器主机名，默认为本地主机(localhost)。
  - user，连接数据库的用户名，默认为当前用户。
  - passwd，连接密码，没有默认值。
  - db，连接的数据库名，没有默认值。
  - conv，将文字映射到Python类型的字典。默认为MySQLdb.converters.conversions
  - cursorclass，cursor()使用的种类，默认值为MySQLdb.cursors.Cursor。
  - compress，启用协议压缩功能。
  - named_pipe，在windows中，与一个命名管道相连接。
  - init_command，一旦连接建立，就为数据库服务器指定一条语句来运行。
  - read_default_file，使用指定的MySQL配置文件。
  - read_default_group，读取的默认组。
  - unix_socket，在unix中，连接使用的套接字，默认使用TCP。
  - port，指定数据库服务器的连接端口，默认是3306。
- 连接对象的db.close()方法可关闭数据库连接，并释放相关资源。

- 连接对象的db.cursor([cursorClass])方法返回一个指针对象，用于访问和操作数据库中的数据。
- 连接对象的db.begin()方法用于开始一个事务，如果数据库的AUTOCOMMIT已经开启就关闭它，直到事务调用commit()和rollback()结束。
- 连接对象的db.commit()和db.rollback()方法分别表示事务提交和回退。
- 指针对象的cursor.close()方法关闭指针并释放相关资源。
- 指针对象的cursor.execute(query[,parameters])方法执行数据库查询。
- 指针对象的cursor.fetchall()可取出指针结果集中的所有行，返回的结果集一个元组(tuples)。
- 指针对象的cursor.fetchmany([size=cursor.arraysize])从查询结果集中取出多行，我们可利用可选的参数指定取出的行数。
- 指针对象的cursor.fetchone()从查询结果集中返回下一行。
- 指针对象的cursor.arraysize属性指定由cursor.fetchmany()方法返回行的数目，影响fetchall()的性能，默认值为1。
- 指针对象的cursor.rowcount属性指出上次查询或更新所发生行数。-1表示还没开始查询或没有查询到数据。

7.3. 模块功能演示

```
#!/usr/bin/python
import MySQLdb

try:
   connection = MySQLdb.connect(user="user",passwd="password",host="xxx",db="test")
except:
   print "Could not connect to MySQL server."
   exit( 0 )

try:
   cursor = connection.cursor()
   cursor.execute( "SELECT note_id,note_detail FROM note where note_id = 1" )
   print "Rows selected:", cursor.rowcount

   for row in cursor.fetchall():
       print "note : ", row[0], row[1]
   cursor.close()
```

Chapter 8. Tkinter模块

8.1. Tkinter简介

Tkinter是Python默认的图形界面接口，Tkinter是一个和Tk接口的Python模块，Tkinter库提供了对Tk API的接口，它属于Tcl/Tk的GUI工具组。Tcl/Tk是由John Ousterhout发展的书写和图形设备。Tcl(工具命令语言)是个宏语言，用于简化shell下复杂程序的开发，Tk工具包是和Tcl一起开发的，目的是为了简化用户接口的设计过程。Tk工具包由许多不同的小部件，如一个按钮、一个滚动条等。通过Tk提供的这些小部件，我们就可快速地进行GUI开发。Perl、Scheme等语言也利用Tk库进行GUI开发。Tkinter是跨平台，在各种平台下都能使用。

Chapter 9. PyGTK

PyGTK是一个用于python GUI程序开发的GTK+库，当前版本的PyGTK需要GTK+ 2.0以上版本支持和Python 2.2以上版本支持才能运行。

9.1. 安装

如果是在Debian系统中，则安装python2.3-gtk2软件包即可。如果要从源码安装，可到http://www.pygtk.org下载最新的软件包。安装方法也很简单，和其它开源软件差不多，通过configure、make和make install三步操作就可完成。具体操作你可参考源码目录下的README和INSTALL文档，里面有详细的安装说明。注意，要成功安装PyGTK，要有相应版本的GTK+和Python支持。在源码目录下有一个examples目录，这是一个宝贵的资源，里面有很多有用的PyGTK示例代码，对我们学习PyGTK很有帮助。

9.2. 示例

下面是一个PyGTK的示例，演示了PyGTK的基本概念。

```
#!/usr/bin/env python
#-*- encoding:utf-8 -*-

import pygtk
pygtk.require('2.0')
import gtk

class base:
#destroy信号的回调函数
        def destroy(self,widget,data=None):
                gtk.main_quit()

#clicked信号的回调函数
        def hello(self,widget,data):
                print 'hello ' + data + ' this is a button clicked() test'

#delete_event事件的回调函数
        def delete_event(self, widget, event, data=None):
                print "delete event occurred"
#如果delete_event事件返回假，则会触发destroy信号，从而关闭窗口。
#如果返回真，则不会关闭窗口。这个特性在当我们需要一个确认是否退出的选择对话框时是很有用。
                return gtk.FALSE

        def __init__(self):
                self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
#设置窗口的delete_event信号触发delete_event函数
                self.window.connect("delete_event", self.delete_event)
#设置窗口的destroy信号触发destroy函数
                handler1 = self.window.connect("destroy",self.destroy)
                print "handler1 is:%d" % handler1
                self.window.set_title('PyGTK 测试 window')
                self.window.set_default_size(200,200)
                self.window.set_border_width(100)
#控制窗口出现的位置
                self.window.set_position(gtk.WIN_POS_CENTER)
#生成按钮实例
                self.button1 = gtk.Button()
                self.button2 = gtk.Button()
                self.button1.set_label('label1')
                self.button2.set_label('label2')
#设置按钮的clicked信号触发hello函数，并传递‘pyGTK’字符串参数给hello函数
                handler2 = self.button1.connect("clicked",self.hello,"pyGTK")
                print "handler2 is:%d" % handler2
#设置按钮的clicked信号触发self..window对象的gtk.Widget.destroy方法
                self.button1.connect_object("clicked", gtk.Widget.destroy, self.window)
#取消handler2的功能
#               self.button.disconnect(handler2)
#设置一个不可见的横向的栏位self.box1
                self.box1 = gtk.HBox(gtk.FALSE, 0)
#把box1放到窗口中
                self.window.add(self.box1)
#把button1部件放到box1中
                self.box1.pack_start(self.button1,gtk.TRUE,gtk.TRUE,0)
                self.button1.show()
#把button2部件放到button1部件之后
                self.box1.pack_start(self.button2,gtk.TRUE,gtk.TRUE,0)
                self.button2.show()
                self.box1.show()
                self.window.show()

        def main(self):
                gtk.main()

print __name__
if __name__ == "__main__":
        base = base()
        base.main()
```

有关PyGTK的详细介绍请参考我整理的“PyGTK学习笔记”。

Chapter 10. PyQt

PyQt是一套用于python的Qt开发库，由一系列的模块组成，有qt, qtcanvas, qtgl, qtnetwork, qtsql, qttable, qtui and qtxml，包含有300个类和超过5750个的函数和方法。

PyQt还支持一个叫qtext的模块，它包含一个QScintilla库。该库是Scintillar编辑器类的Qt接口。

10.1. 安装

到http://www.riverbankcomputing.co.uk/pyqt/download.php下载最新的版本。安装PyQt需要先安装SIP，到以这里下载。SIP是一个把C\C++库转换成Python模块的工具。

- 安装SIP

```
% tar -zxvf sip-4.1.1.tar.gz
% cd sip-4.1.1
% python configure.py -l qt          # -l qt 选项指定qt版本
% make
% make install
```

- 安装PyQt

```
% tar -zxvf PyQt-x11-gpl-3.13.tar.gz
% cd PyQt-x11-gpl-3.13
% python configure.py
% make
% make install
```

Chapter 11. PyMedia

PyMedia模块是一个用于多媒体操作的python模块。它提供了丰富而简单的接口用于多媒体处理(wav, mp3, ogg, avi, divx, dvd, cdda etc)。可在Windows和Linux平台下使用。

Chapter 12. Python-ldap

Python-ldap模块提供一组面向对象的API，可方便地在python中访问ldap目录服务，它基于OpenLDAP2.x。

12.1. 示例

- 以下示例在python-ldap网站上有列出，这里作一下简要说明：

```
#!/usr/bin/python
#-*- coding:utf-8 -*-                              #设置源码文件编码为utf-8

import ldap

try:
   conn = ldap.open("server_name")                 #server_name为ldap服务器名
   conn.protocol_version = ldap.VERSION3           #设置ldap协议版本
   username = "cn=admin,dc=company,dc=com"         #用户名
   password = "123"                                #访问密码
   conn.simple_bind(username,password)             #连接

except ldap.LDAPError, e:                          #捕获出错信息
   print e

baseDN = "dc=employees,dc=company,dc=com"          #设置目录的搜索路径起点
searchScope = ldap.SCOPE_SUBTREE                   #设置可搜索子路径

retrieveAttributes = None                          #None表示搜索所有属性，['cn']表示只搜索cn属
searchFilter = "cn=test"                           #设置过滤属性，这里只显示cn=test的信息

try:
   ldap_result_id = conn.search(baseDN,searchScope,searchFilter,retrieveAttributes)
#调用search方法返回结果id
   result_set = []
while 1:
   result_type, result_data = conn.result(ldap_result_id, 0)       #通过结果id返回信息
   if result_data == []:
      break
   else:
      if result_type == ldap.RES_SEARCH_ENTRY:
         result_set.append(result_data)

   print result_set[0][0][1]['o'][0]       #result_set是一个复合列表，需通过索引返回组织单

except ldap.LDAPError, e:
   print e
```

这里采用的是非同步方式，同步方式的连接和搜索命令后有“_s”后缀，如search_s。非同步方式需通过一个结果id来访问目录服务信息。

- 下面是一个修改目录信息的示例：

```
#!/usr/bin/python
# -*- coding:utf-8 -*-
import ldap

try:
   conn = ldap.open("server_name")
   conn.protocol_version = ldap.VERSION3
   username = "cn=admin,dc=company,dc=com"
   password = "123"
   conn.simple_bind_s(username,password)

except ldap.LDAPError, e:
   print e

try:
   dn = "cn=test,dc=employees,dc=company,dc=com"
   conn.modify_s(dn,[(ldap.MOD_ADD,'mail','test@163.com')])      #增加一个mail属性
except ldap.LDAPError, e:
   print e
```

ldap.MOD_ADD表示增加属性，ldap.MOD_DELETE表示删除属性，ldap.MOD_REPLACE表示修改属性。

- 下面是一个增加目录项的示例：

```
#!/usr/bin/python
# -*- coding:utf-8 -*-
import ldap,ldap.modlist                 #ldap.modlist是ldap的子模块，用于格式化目录服务的

try:
        conn = ldap.open("server_name")
        conn.protocol_version = ldap.VERSION3
        username = "cn=admin,dc=company,dc=com"
        password = "123"
        conn.simple_bind_s(username,password)

except ldap.LDAPError, e:
        print e

try:
        dn = "cn=test,dc=card,dc=company,dc=com"
        modlist = ldap.modlist.addModlist({          #格式化目录项，除对象类型要求必填项外
        'cn': ['test'],                              #其它项可自由增减
        'objectClass': ['top', 'person', 'organizationalPerson', 'inetOrgPerson'],
        'o': ['\xe5\xb9\xbf\xe5\xb7\x9e'],          #这些为utf-8编码的中文
        'street': ['\xe5\xb9\xbf\xe5\xb7\x9e'],
        'sn': ['tester'],
        'mail': ['test@163.com', 'test@21cn.com'],
        'homePhone': ['xxxxxxxx'], 'uid': ['test'] })
#       print modlist                                #显示格式化数据项，格式化后是一个元组
        conn.add_s(dn,modlist)                       #调用add_s方法添加目录项

except ldap.LDAPError, e:
        print e
```

其实我们也可按格式化后元组列表的形式把目录项直接写到add_s()里，省却转换的步骤。

- 下面是一个删除目录项的示例：

```
#!/usr/bin/python
# -*- coding:utf-8 -*-
import ldap

try:
        conn = ldap.open("server_name")
        conn.protocol_version = ldap.VERSION3
        username = "cn=admin,dc=company,dc=com"
        password = "123"
        conn.simple_bind_s(username,password)

except ldap.LDAPError, e:
        print e

try:
        dn = "cn=sale,dc=company,dc=com"
        conn.delete_s(dn)

except ldap.LDAPError, e:
        print e
```

Chapter 13. ftplib -- FTP protocol client

ftplib模块定义了FTP类和一些方法，用以进行客户端的ftp编程。我们可用python编写一个自己的ftp客户端程序，用于下载文件或镜像站点。如果想了解ftp协议的详细内容，请参考RFC959。

13.1. 示例

该模块是python的通用模块，所以默认应该已安装。ftplib模块使用很简单，暂时只有一个FTP类和十几个函数。下面用一个交互方式演示一下ftplib的主要功能。

```
>>> from ftplib import FTP
>>> ftp=FTP('ftp.python.org')
>>> ftp.login()
'230 Login successful.'
>>> ftp.dir()
drwxrwxr-x    7 1004     1004          512 Aug 13 01:35 pub
>>> ftp.cwd('pub')
'250 Directory successfully changed.'
>>> ftp.dir()
drwxrwxr-x    5 1000     1004         1024 Dec 24 11:04 docs.python.org
drwxrwsr-x    2 1002     1004          512 Oct 12  2001 jython
lrwx------    1 0        1003           25 Aug 03  2001 python -> www.python.org/ftp/pyth
drwxr-xr-x    9 1018     1004          512 Feb 02 03:44 pyvault
drwxr-xr-x    2 1005     1004          512 May 06  2003 tmp
drwxrwsr-x   59 1004     1004         3072 Feb 03 14:58 www.python.org
>>> ftp.quit()
'221 Goodbye.'
```

下面一个下载文件的示例

```
#!/usr/bin/env python

#author:Jims of www.ringkee.com
#create date: 2005/02/05
#description: Using ftplib module download a file from a ftp server.

from ftplib import FTP

ftp=FTP()

ftp.set_debuglevel(2)                 #打开调试级别2，显示详细信息
ftp.connect('ftp_server','port')      #连接
ftp.login('username','password')      #登录，如果匿名登录则用空串代替即可

print ftp.getwelcome()                #显示ftp服务器欢迎信息
ftp.cwd('xxx/xxx/')                   #选择操作目录
bufsize = 1024                        #设置缓冲块大小
filename='dog.jpg'
file_handler = open(filename,'wb').write            #以写模式在本地打开文件
ftp.retrbinary('RETR dog.jpg',file_handler,bufsize)  #接收服务器上文件并写入本地文件
ftp.set_debuglevel(0)                               #关闭调试

ftp.quit()                                          #退出ftp服务器
```

下面一个上传文件的示例，要成功运行该脚本，需在ftp服务器上有上传文件的权限。

```
#!/usr/bin/env python

#author:Jims of www.ringkee.com
#create date: 2005/02/05
#description: Using ftplib module upload a file to a ftp server.

from ftplib import FTP

ftp=FTP()

ftp.set_debuglevel(2)
ftp.connect('ftp_server','port')
ftp.login('username','password')

print ftp.getwelcome()
ftp.cwd('xxx/xxx/')
bufsize = 1024
filename='dog.jpg'
file_handler = open(filename,'rb')
ftp.storbinary('STOR dog.jpg',file_handler,bufsize)   #上传文件
ftp.set_debuglevel(0)

file_handler.close()         #关闭文件
ftp.quit()
```

是不是很简单，其它功能可查询python的官方网站，网址是http://docs.python.org/lib/module-ftplib.html。

Chapter 14. Psyco

Psyco是一个Python代码加速度器，可使Python代码的执行速度提高到与编译语言一样的水平。

14.1. 安装

安装Psyco很简单，它有两种安装方式，一种是源码方式，一种是二进制码方式：

- 如果用源码方式安装，你需在源码的目录中调用python setup.py install命令编译生成psyco子目录，再把该子目录整个拷贝到python的site-packages目录下。
- 如果用二进制码方式安装，按这个网址列表中的python与psyco版本对应表下载合适的二进制文件，解压后会生成一个psyco-1.x的目录，把该目录下的psyco目录整个拷贝到python的site-packages目录下即可。

Chapter 15. smtplib

15.1. 示例

smtplib模块以发送电子邮件。下面是一个示例，可发送附件。

```
#!/usr/bin/python
#-*- encoding:utf-8 -*-

import smtplib,mimetypes
from email import Encoders
from email.MIMEBase import MIMEBase
from email.MIMEText import MIMEText
from email.MIMEMultipart import MIMEMultipart

msg = MIMEMultipart()                        #创建可包含附件的MIME对象
msg['Subject'] = 'this is title'
msg['From'] = 'yjnet@21cn.com'
msg['To'] = 'yjnet@21cn.com'

txt = MIMEText('这是邮件正文的中文测试。',_charset='utf-8')
msg.attach(txt)

filename = 'jdk15.tar'                       #附件名
fp = open(filename,'rb')
ctype,encoding = mimetypes.guess_type(filename)
if ctype is None or encoding is not None:
        ctype = 'application/octet-stream'
maintype,subtype = ctype.split('/',1)
m = MIMEBase(maintype,subtype)
m.set_payload(fp.read())
fp.close()
Encoders.encode_base64(m)                                               #把附件编码
m.add_header('Content-disposition','attachment',filename=filename)    #修改邮件头
msg.attach(m)                                                           #添加附件

s = smtplib.SMTP('smtp.21cn.com')                                       #连接邮件服务器
s.login('yjnet','****')                                                 #登录邮件服务器
s.sendmail('yjnet@21cn.com','yjnet@21cn.com',msg.as_string())          #发送邮件
s.close()
```

Chapter 16. XMPPPY

Jabber服务器采用开发的XMPP协议，Google Talk也是采用XMPP协议的IM系统 。在Python中有一个xmpppy模块支持该协议。也就是说，我们可以通过该模块与Jabber服务器通信，是不是很Cool。

16.1. 示例

下面是一个简单的示例，可使大家对该模块有一个大概的了解。

```
#导入xmpppy模块
>>> import xmpp
#建立Client实例，debian是我的jabber服务器名，jabber服务器的安装可参考我的Debian学习笔记。
>>> c=xmpp.Client('debian',debug=[])
#连接
>>> c.connect()
'tcp'
#验证
>>> c.auth('yangjing','12345')
'old_auth'
#登入
>>> c.sendInitPresence()
#向ringkee@debian
>>> c.send(xmpp.protocol.Message('ringkee@debian ','test message from yangjing'))
'20'
#下面测试信息接收功能，如果没有信息，则pending_data()为空
>>>c.TCPsocket.pending_data()
[]
#如果有信息，则pending_data()不为空
>>> c.TCPsocket.pending_data()
[<socket._socketobject object at 0xb795beb4>]
#接收信息
>>> c.TCPsocket.receive()
"<message type='chat' to='yangjing@debian/xmpppy' from='ringkee@debian/Gaim'><x xmlns='ja
#登出
>>> c.disconnect()
```

16.2. cjkcodecs

在python2.4版以前，python不能处理cjk（中国，日本和韩国）的编码，所以就有了cjkcodecs模块。安装该模块后Python就能处理cjk字符了。下载网址：http://cjkpython.i18n.org/ 。

Sed 学习笔记

作者：Jims of 肥肥世家

jims.yang@gmail.com

发布时间:2004年09月20日

最近更新:2005年12月22日，增加小技巧章节。

Table of Contents

1. Sed简介

sed是一种在线编辑器，它一次处理一行内容。处理时，把当前处理的行存储在临时缓冲区中，称为“模式空间”（pattern space），接着用sed命令处理缓冲区中的内容，处理完成后，把缓冲区的内容送往屏幕。接着处理下一行，这样不断重复，直到文件末尾。文件内容并没有改变，除非你使用重定向存储输出。Sed主要用来自动编辑一个或多个文件；简化对文件的反复操作；编写转换程序等。以下介绍的是Gnu版本的Sed 3.02。

2. 定址

可以通过定址来定位你所希望编辑的行，该地址用数字构成，用逗号分隔的两个行数表示以这两行为起止的行的范围（包括行数表示的那两行）。如1，3表示1，2，3行，美元符号($)表示最后一行。范围可以通过数据，正则表达式或者二者结合的方式确定 。

3. Sed命令

调用sed命令有两种形式：

- `sed [options] 'command' file(s)`
- `sed [options] -f scriptfile file(s)`

```
a\
在当前行后面加入一行文本。
b lable
分支到脚本中带有标记的地方，如果分支不存在则分支到脚本的末尾。
c\
用新的文本改变本行的文本。
d
从模板块（Pattern space）位置删除行。
D
删除模板块的第一行。
i\
在当前行上面插入文本。
h
拷贝模板块的内容到内存中的缓冲区。
H
追加模板块的内容到内存中的缓冲区
g
获得内存缓冲区的内容，并替代当前模板块中的文本。
G
获得内存缓冲区的内容，并追加到当前模板块文本的后面。
l
列表不能打印字符的清单。
n
读取下一个输入行，用下一个命令处理新的行而不是用第一个命令。
N
追加下一个输入行到模板块后面并在二者间嵌入一个新行，改变当前行号码。
p
打印模板块的行。
P（大写）
打印模板块的第一行。
q
退出Sed。
```

```
r file
从file中读行。
t label
if分支，从最后一行开始，条件一旦满足或者T，t命令，将导致分支到带有标号的命令处，或者到脚本的末尾。
T label
错误分支，从最后一行开始，一旦发生错误或者T，t命令，将导致分支到带有标号的命令处，或者到脚本的末尾。
w file
写并追加模板块到file末尾。
W file
写并追加模板块的第一行到file末尾。
!
表示后面的命令对所有没有被选定的行发生作用。
s/re/string
用string替换正则表达式re。
=
打印当前行号码。
#
把注释扩展到下一个换行符以前。
```

以下的是替换标记

- `g` 表示行内全面替换。
- `p` 表示打印行。
- `w` 表示把行写入一个文件。
- `x` 表示互换模板块中的文本和缓冲区中的文本。
- `y` 表示把一个字符翻译为另外的字符（但是不用于正则表达式）

4. 选项

```
-e command, --expression=command
```

允许多台编辑。

```
-h, --help
```

打印帮助，并显示bug列表的地址。

```
-n, --quiet, --silent
```

取消默认输出。

```
-f, --filer=script-file
```

引导sed脚本文件名。

```
-V, --version
```

打印版本和版权信息。

5. 元字符集

```
^
锚定行的开始 如：/^sed/匹配所有以sed开头的行。
$
锚定行的结束 如：/sed$/匹配所有以sed结尾的行。
.
匹配一个非换行符的字符 如：/s.d/匹配s后接一个任意字符，然后是d。
*
匹配零或多个字符 如：/*sed/匹配所有模板是一个或多个空格后紧跟sed的行。
[]
匹配一个指定范围内的字符，如/[Ss]ed/匹配sed和Sed。
[^]
匹配一个不在指定范围内的字符，如：/[^A-RT-Z]ed/匹配不包含A-R和T-Z的一个字母开头，紧跟ed的行。
\(..\)
保存匹配的字符，如s/\(love\)able/\1rs，loveable被替换成lovers。
&
保存搜索字符用来替换其他字符，如s/love/**&**/，love这成**love**。
\<
锚定单词的开始，如:/\&lt;love/匹配包含以love开头的单词的行。
\>
锚定单词的结束，如/love\&gt;/匹配包含以love结尾的单词的行。
x\{m\}
重复字符x，m次，如：/0\{5\}/匹配包含5个o的行。
x\{m,\}
重复字符x,至少m次，如：/o\{5,\}/匹配至少有5个o的行。
x\{m,n\}
重复字符x，至少m次，不多于n次，如：/o\{5,10\}/匹配5--10个o的行。
```

6. 实例

删除：d命令

- `$ sed '2d' example` -----删除example文件的第二行。
- `$ sed '2,$d' example` -----删除example文件的第二行到末尾所有行。
- `$ sed '$d' example` -----删除example文件的最后一行。
- `$ sed '/test/'d example` -----删除example文件所有包含test的行。

替换：s命令

- `$ sed 's/test/mytest/g' example` -----在整行范围内把test替换为mytest。如果没有g标记，则只有每行第一个匹配的test被替换成mytest。
- `$ sed -n 's/^test/mytest/p' example` -----(-n)选项和p标志一起使用表示只打印那些发生替换的行。也就是说，如果某一行开头的test被替换成mytest，就打印它。
- `$ sed 's/^192.168.0.1/&localhost/' example` -----&符号表示替换换字符串中被找到的部份。所有以192.168.0.1开头的行都会被替换成它自已加localhost，变成192.168.0.1localhost。
- `$ sed -n 's/\(love\)able/\1rs/p' example` -----love被标记为1，所有loveable会被替换成lovers，而且替换的行会被打印出来。
- `$ sed 's#10#100#g' example` -----不论什么字符，紧跟着s命令的都被认为是新的分隔符，所以，“#”在这里是分隔符，代替了默认的“/”分隔符。表示把所有10替换成100。

选定行的范围：逗号

- `$ sed -n '/test/,/check/p' example` -----所有在模板test和check所确定的范围内的行都被打印。
- `$ sed -n '5,/^test/p' example` -----打印从第五行开始到第一个包含以test开始的行之间的所有行。
- `$ sed '/test/,/check/s/$/sed test/' example` -----对于模板test和west之间的行，每行的末尾用字符串sed test替换。

多点编辑：e命令

- `$ sed -e '1,5d' -e 's/test/check/' example` -----(-e)选项允许在同一行里执行多条命令。如例子所示，第一条命令删除1至5行，第二条命令用check替换test。命令的执行顺序对结果有影响。如果两个命令都是替换命令，那么第一个替换命令将影响第二个替换命令的结果。
- `$ sed --expression='s/test/check/' --expression='/love/d' example` -----一个比-e更好的命令是--expression。它能给sed表达式赋值。

从文件读入：r命令

- `$ sed '/test/r file' example` -----file里的内容被读进来，显示在与test匹配的行后面，如果匹配多行，则file的内容将显示在所有匹配行的下面。

写入文件：w命令

- `$ sed -n '/test/w file' example` -----在example中所有包含test的行都被写入file里。

追加命令：a命令

- `$ sed '/^test/a\\--->this is a example' example<` -----'this is a example'被追加到以test开头的行后面，sed要求命令a后面有一个反斜杠。

插入：i命令

```
$ sed '/test/i\\

new line

-------------------------' example
```

如果test被匹配，则把反斜杠后面的文本插入到匹配行的前面。

下一个：n命令

- `$ sed '/test/{ n; s/aa/bb/; }' example` -----如果test被匹配，则移动到匹配行的下一行，替换这一行的aa，变为bb，并打印该行，然后继续。

变形：y命令

- `$ sed '1,10y/abcde/ABCDE/' example` -----把1--10行内所有abcde转变为大写，注意，正则表达式元字符不能使用这个命令。

退出：q命令

- `$ sed '10q' example` -----打印完第10行后，退出sed。

保持和获取：h命令和G命令

- `$ sed -e '/test/h' -e '$G example` -----在sed处理文件的时候，每一行都被保存在一个叫模式空间的临时缓冲区中，除非行被删除或者输出被取消，否则所有被处理的行都将打印在屏幕上。接着模式空间被清空，并存入新的一行等待处理。在这个例子里，匹配test的行被找到后，将存入模式空间，h命令将其复制并存入一个称为保持缓存区的特殊缓冲区内。第二条语句的意思是，当到达最后一行后，G命令取出保持缓冲区的行，然后把它放回模式空间中，且追加到现在已经存在于模式空间中的行的末尾。在这个例子中就是追加到最后一行。简单来说，任何包含test的行都被复制并追加到该文件的末尾。

保持和互换：h命令和x命令

- `$ sed -e '/test/h' -e '/check/x' example` -----互换模式空间和保持缓冲区的内容。也就是把包含test与check的行互换。

7. 脚本

Sed脚本是一个sed的命令清单，启动Sed时以-f选项引导脚本文件名。Sed对于脚本中输入的命令非常挑剔，在命令的末尾不能有任何空白或文本，如果在一行中有多个命令，要用分号分隔。以#开头的行为注释行，且不能跨行。

8. 小技巧

- 在sed的命令行中引用shell变量时要使用双引号，而不是通常所用的单引号。下面是一个根据name变量的内容来删除named.conf文件中zone段的脚本：

```
name='zone\ "localhost"'
sed "/$name/,/};/d" named.conf
```

VIM 学习笔记

整理：Jims of 肥肥世家

jims.yang@gmail.com

发布时间:2005年07月18日

更新时间:2006年05月22日

Abstract

VI编辑器是类UNIX系统中最常用到的编辑器，VIM是VI编辑器的改良版本。本笔记是我在使用vim时记录下来的备忘录，内容不是很全面，详细介绍可参考http://vimdoc.sourceforge.net/网站。

Table of Contents

1. 普通模式

在shell中直接打vim filename就会进入普通模式。在这个状态，我们可以通过不同的按键对文件进行操作和切换到其它模式。按“:”可切换到命令模式，按i，o，a可进入编辑模。

1.1. 编辑

`i`

进入编辑模式。

`o`

在当前位置下插入一空行，进入编辑模式，光标位于空行的最开头。

`a`

光标后移一个字符，进入编辑模式。

`v`

这入可视模式，可用高亮的色块选择内容。

`x or DEL键`

删除当前字符，删除内容保存在缓冲区。

`xp`

左右字符互换。

`dd`

删除当前行，删除内容保存在缓冲区。

`ddp`

上下两行的内容互换。

`d$`

删除当前光标至行尾的所有内容。

`dG`

删除从当前行至文件未尾的所有行。

`u`

undo。

`v`

进地visual模式，移动光标可选择文本。

`y`

把当前行复制到缓冲区中。

`p`

把缓冲区中的文本插入到当前位置。

`"ay`

把当前行复制到a缓冲区。可用26个字母命名多个缓冲区。

`"ap`

把a缓冲区中的文本插入当前位置。

`J`

上下两行合并成一行。

`.`

英文句点的作用是重复执行上次执行的命令，如你按了"ap插入a缓冲区的内容，那你就可按"."来重复这个操作。

1.2. 光标移动

`h，j，k，l`

在vim中，除了可使用光标键在移动光标外，还有一种更方便的光标移动方式。就是使用h，j，k，l这四个键来移动光标。h控制光标左移，j控制光标下移，k控制光标上移，l控制光标右移。通过使用这四个字母键就可使我们的手不用移动即可控制光标的移动。刚开始使用可能会有些不习惯，但熟练使用后你会发觉你的输入速度提高不少，强烈建议喜欢vim的朋友使用。

`$`

光标移动到行尾。

`G`

光标移动到文档末尾。

`H,L`

H控制光标移动到当前屏幕头，L控制光标移动到当前屏幕尾。

`{，}`

{控制光标上移一个段落，}控制光标下移一个段落

2. 编辑模式

`ESC`

退出编辑状态。

3. 命令模式

在普通模式上按“:”就可进入命令模式，在左下屏幕我们输入一些操作指令。

`:q!`

不保存退出vim。

`:w`

保存文档，但不退出vim。

`:x`

保存退出vim。

`:! command`

运行shell命令。

`:e filename`

编辑/打开一个文件

`:s/emacs/vim`

在当前行中把第一个emacs替换成vim。

`:s/emacs/vim/g`

把当前行中所有的emacs替换成vim。

`:%s/emacs/vim/g`

在全局范围内把emacs替换成vim。

`:reg`

列出缓冲区内容。

`:set all`

列出所有参数的配置情况。

`:tabe`

新建一个标签页。

`:tabn or :tabp`

切换到下一个\上一个标签页。

`:close`

关闭当前标签页。

`:qa`

关闭所有标签页退出。

4. vimrc配置

在命令模式下用set命令设置的东西是不能保存的，下次打开vim时又要重新设置。所以vim提供了一个配置文件叫vimrc，可以保存你的配置信息。该文件在Debian系统中位于/etc/vim/目录下。在该文件中，以双引号开头的是注释。

- `set autoindent`

自动缩排，如当前行是从第3个字符的位置开始编辑的,按回车后光标会自动定位在下一行第三3个字符的位置。

- `set paste`

  置粘贴模式，这样粘贴过来的程序代码就不会错位了。

- 打开文件时自动回到上次编辑位置。

```
if has("autocmd")
   autocmd BufRead *.txt set tw=78
   autocmd BufReadPost *
      \ if line("'\"") &gt; 0 && line ("'\"") &lt;= line("$") |
      \   exe "normal g'\"" |
      \ endif
endif
```

XML 学习笔记

整理：Jims of 肥肥世家

jims.yang@gmail.com

发布时间:2004年12月03日

最近更新:2006年02月17日，新增CSS内容

Abstract

XML技术是Internet技术的又一次革命。本笔记记录标记语言的历史和发展，还有我的学习历程。

Table of Contents

Chapter 1. XML简介

XML(eXtensible Markup Language，可扩展标记语言)是SGML的一个子集，但比SGML简单，用以创建可相互转换的结构化文本文档和数据文档。下面说明一下与XML相关的一些概念。

- SGML(Standard Generalized Markup Language，标准通用标记语言)，由于IBM公司的三位先驱者Charles GoldFarb、Edward Mosher和Raymond Lorie创立，主要作为大型文档的编制工具。DTD(Document Type Definition，文档类型定义)是SGML文档的核心，它定义了SGML文档必须遵循的一组语法规则。由于它很复杂，所以只是在一些大公司或大项目中使用。直到HTML面世，它还是默默无闻。

- HTML(Hypertext Markup Language，超文本标记语言)，它是在SGML框架中通过DTD定义的标记语言，是SGML的一种应用。它由于结构简单，容易学习而迅速普及，每个人都能很快地建立自已的页面，HTML造就了现时Internet上无数的信息资源。HTML标记只描述文档的外观，而不描述文档的内容本身--里面有什么。HTML是不明白网页内容的，这样就造成了内容搜索的差异和不确定性。另一个问题是，HTML不是可扩展的，这意味着没有一种方便的途径来扩展标记。每一个新标记的引入都会造成系统的不一致性和对标准的修订。这就是为什么现在我们用不同的浏览器浏览同一个网站时表现效果会有差异。

- XHTML(eXtensible Hypertext Markup Language，可扩展超文本标记语言)，它是按XML规则编写的HTML，由于有统一的规则约束，所以它不会出现如HTML一样的不规范、不一致性问题。

- XML(eXtensible Markup Language，可扩展标记语言)，继承了SGML的优点，但又没有了SGML的复杂性。XML专门为WEB应用而设计，和HTML不同，它是一种元标记语言(meta-markup language)，也就是说它没有一套能够适用于各个领域中所有用户的固守的标签和元素，相反，它允许开发者根据自已的需要定义自已的元素，XML中的X(eXtensible)就是说明了这一点。它的特点有：

- XML使用Unicode字符集，可生成英文、中文、希腊文或梵文等多种语言。
- 可将多个来源(包括其他XML文档和二进制文件)汇合进一个XML文档。
- 可利用DTD或Schema(模式）管理一致性问题。DTD主要用于文档型文档，Schema主要用数据型文档。
- 具有很好的扩展性，可定义自已的元素和属性。
- 通过XML可从关系数据库管理系统中提取数据到结构化文档。它还被设计成可对各种数据对象进行操作。
- 在一个设计良好的XML应用中，XML标记不涉及文档如何显示，只表示文档的结构。

> XML被设计用来存储、支持和交换数据，而不是用来显示数据的。通常，XML被用于数据交换，而不是数据存储。

- 元数据，定义数据的数据。
- 标记语言是一种定义文档的格式语言。SGML、XML、XHML、HTML都属标记语言。

XML文档是什么？它有时是一个文件，有时是关系数据库中的一条记录，有时是由Object Request Broker(对象请求代理程序)传送的一个对象，有时是到达网络接口的一个字节流。XML文档可使不同系统、不同平台的数据实现统一接口，这就是XML眞正的威力所在。下面列举几个使用XML的领域：

- 文档设计和管理，可利用XML维护公司的文档资料。
- Web开发，利用XHTML和XSLT实现的Web页面扩展性更好，更容易维护。
- 数据库应用和程序开发，可从数据库中提取数据并生成XML文档，实现信息的跨平台、跨系统沟通。
- 定义其它语言，WML和WAP就是用通过XML建立的。

XML不是什么？

- XML只是一种标记语言，不是一种编程语言。不存在一种编译器，把XML文档转化成可执行二进制代码。
- XML不是一种网络传输协议，但通过网络协议传输的数据格式则可以是XML格式的。
- XML不是数据库，不能替代Oracle或MySQL这类的关系数据库管理系统。

Chapter 2. XML语法

创建一个简单的index.xml文档：

```
< xml version="1.0" >
< xml-stylesheet type="text/xsl" href="basic.xsl" >
<basic>Hello World</basic>
```

下面创建一个名为basic.xsl的XML样式表(XSL)，以便在浏览器中显示XML文档内容：

```
< xml version="1.0" >
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
        <html>
           <head>
           <title>a basic stylesheet</title>
           </head>
           <body>
           <xsl:value-of select="/" />
           </body>
        </html>
</xsl:template>
</xsl:stylesheet>
```

接着在浏览器中打开index.xml文档，则可显示“Hello World”。上面两个文档都是合法的XML文件，具体的语法规则下面会详细介绍，上例可先给大家一个感性的认识。

合法的XML文档可有种意思，一个是良构文档(well-format)，即符合XML规则书写的文档；另一种是有效文档，是已验证符合一个DTD的文档。

2.1. 基本语法规则

- XML是区分大小写的；
- 所有元素的起始和结束标注必须成对出现，且要正确嵌套；
- 如果使XML说明，则它必须是XML文档的第一行：

  ```
  < xml version="1.0" >
  ```

- 元素属性必须用引号引起来，单、双引号都可以，但必须成对出现。如：

  ```
  <basic attr="1.0">
  <basic attr='1.0'>
  ```

- XML命名规则：
  - XML名以下划线或字母开始；
  - XML名可包含字母、数字、句点、下划线和冒号；
  - XML名不能包含空格；
  - XML名不能以数字开始，但可包含数字；

    - XML名区分大小写。

- 保留标记字符，如果要在XML中显示<或&之类的标记，就要使用字符的实体形式，XML中有五种预先定义了的实体：

    ```
    <          表示<字符
    >          表示>字符
    &amp;         表示&字符
    &apos;        表示'字符
    &quot;        表示"字符
    ```

    我们也可用ENTITY自定义实体：

    ```
    <!ENTITY linux "linux is a very good system">
    这样我们可用&linux;来调用。
    ```

- XML文档内容中的空格是有意义的，在转换后会保留。

- 空元素以<开始并以/>结束，如
。

2.2. 良构XML文档和有效XML文档

符合XML语法规则的XML文档称为良构文档，这些规则如下：

- 应当只有一个父标志，由父标志派生所有其它子标志，在一个文档中不能存在多个父标志。

- 嵌套元素应按正确的顺序开始和结束。

- 子标志应在父标志完成前关闭。

- 属性值应放在双引号中。

通过某个DTD或Schema验证的文档称为有效XML文档。

2.3. XML文档的组成

- XML声明：

    - version，定义XML规范的版本号，到现在为止，只有一个版本号1.0。

    - encoding，指定文档的编码系统。

    - standalone，定义文档是独立的还是需要装入其他元素才能正确分析。如果XML文档没有外部实体或DTD，则可以设置为no，否则设置为yes。可用该值提高性能：如果为no，则可提高处理速度；如果设置为yes，则首先要分析文档，确定需要其他哪些文件，然后才能完全分析文档。

- 根元素，每篇XML文档都需要有且只能有一个根元素。由元素是文档的第一个元素，包含其它所有元素。下例的portal就是根元素，如：

```
<portal>
 <name>jims</name>
 <email></email>
 ...
</portal>
```

- 属性，每个元素都可以设置一个或多个属性，如：

```
<portal>
  <name id='1',sex="male">Jims</name>
</portal>
```

元素和属性都可以表示信息，什么时候使用元素，什么时候使用属性呢？属性信息表现能力有限，它只能表示字符串。所以当需灵活表示信息时应该使用元素。一般把信息主体放到元素中，属性只放一些注释或额外的信息。

- CDATA部份，它用<![CDATA[和]]>表示，它们之间的数据作为原始字符显示，唯一不能出现的标志是]]>。
- 注释，注释是很重要，不论是在编写程序和文档时，所以XML也提供了注释功能，以<!--开头-->结尾的一对区间为注释。在以-->结束之前，不能出现“--”号，“---”更不允许。
- 处理指令，处理指令以< 开头以 >结尾。如PHP处理指令可写成，< php ... >。处理指令是标记，而不是元素。因此，与注释一样，处理指令可出现在XML文档的标签外的任何位置，包括根元素之前或之后。最常见的处理指令是，xml-stylesheet样式表指令，它会告诉浏览器在显示文档时应用什么样式表。如：

```
< xml-stylesheet href="sample.css" type="text/css" >
<portal>
  <name>...</name>
...
</portal>
```

2.4. XML文档树

XML文档是一种结构化的文档，可用树的形式表示出来。树是一种由节点和分支组成的简单结构，两个节点间由分支连接。上端的节点称为父节点，下端的节点称为子节点。一个节点如果没有父节点，则称为树的根节点(根)，每个树必须有且只能有一个根节点。一个节点如果没有子节点，则称为树的叶节点。只有一个节点的树也是允许的。

Chapter 3. DTD

由于XML可自定义标签，所以每个人定义的标签集都会不同，如果没有一套标准来规定标签的定义原则，则应用程序就不能对XML文档进行处理。解决该问题的方案采用DTD，DTD(Document Type Definition，文档类型定义)，用于定义XML文档的编写规则。如哪些元素可出现在文档中，及元素的内容和属性的要求等。应用程序会利用这个DTD对文档进行检验，符合DTD约束规则的XML文档称之为有效文档，可以进行下一步处理，否则会报错，应用程序可捕获该错误进行相应的异常处理。检验过程是可选，这要视具体应用而定。

3.1. 文档类型声明

要使用DTD进行有效性检验，就要使用文档类型定义声明指定DTD。如：

```
< xml version="1.0" standalone="no" >
<!DOCTYPE portal SYSTEM "http://www.w3c.com/dtd/portal.dtd">
<portal>
  <name>Jims</name>
  <email>Jims@163.com</email>
  <email>Jims@21cn.com</email>
</portal>
```

文档类型声明位于XML声明之后，根元素之前。如果dtd文档位于本机，可用路径名直接指出dtd文档的位置。portal.dtd的内容如下：

```
<!ELEMENT portal (name,email*)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT email (#PCDATA)>
```

上面的内容也可直接写到XML文档内，这种dtd声明方式叫内部dtd子集，如：

```
< xml version="1.0" standalone="no" >
<!DOCTYPE portal [
<!ELEMENT portal (name,email*)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT email (#PCDATA)>
]>
<portal>
  <name>Jims</name>
  <email>Jims@163.com</email>
  <email>Jims@21cn.com</email>
</portal>
```

如果dtd位于XML文档外，则叫外部dtd子集。我们可以结合内外dtd，共同组成一个dtd来为XML文档作验证。如：

```
<!DOCTYPE portal SYSTEM "external.dtd" [
<!ELEMENT portal (name,email*)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT email (#PCDATA)>
]>
```

注意，使用内外dtd时，这两个dtd要互相兼容，不能有冲突。

3.2. 元素声明

上节文档类型声明中的每一项都是元素声明，定义了每个元素的约束。元素声明的格式为：

```
<!ELEMENT element_name (content_model)>
```

有效文档中使用的每个元素都必须在文档的DTD中用元素声明进行声明。element_name可是任何合法的XML名称，content_model(内容模型)指定元素可以或必须包含的子元素以及子元素的顺序。下面具体介绍内容模型的内容。

- # PCDATA，规定元素只包含已析的字符数据。下面声明指出一个name元素可以包含文本，但不能划分为独立的area_code、number和extension元素：

  ```
  <!ELEMENT name (#PCDATA)>
  ```

- 子元素，可指明元素的子元素。下面声明表示name元素必须包含且只包含一个desc元素。

  ```
  <!ELEMENT name (desc)>
  ```

  也可用逗号为分隔符，指明多个子元素。并且子元素出现的次序必须按定义时的顺序。如：

  ```
  <!ELEMENT name (id,desc)>
  ```

  name元素的id子元素必须在desc子元素前面，否则验证会出错，该文档不是一个有效的XML文档。

```
下面这个文档是有效的
<name>
   <id>1</id>
   <desc>dtd test</desc>
</name>
下面这个文档是无效的，顺序颠倒了
<name>
   <desc>dtd test</desc>
   <id>1</id>
</name>
下面的文档也是无效的，有多余的元素
<name>
   <id>1</id>
   <desc>dtd test</desc>
   <date>2005/01/31</date>
</name>
```

- 子元素的个数，我们可通过正则表达式来规定子元素的个数。
  - ，允许零个或一个该元素
  - *，允许零个或多个该元素
  - +，允许一个或多个该元素

下面我们可利用这些符号规定id子元素必须出现，且只能出现一次，而desc子元素可选。

```
<!ELEMENT name (id，desc*)>
```

根据上面的声明，下面的name元素都是有效的。

```
<name>
   <id>1</id>
   <desc>dtd test</desc>
</name>
<name>
   <id>2</id>
</name>
<name>
   <id>3</id>
   <desc>dtd test</desc>
   <desc>another test</desc>
</name>
```

- 可选项(|)，选项是一个参数列表，每个参数间用“|”分隔，代表能且只能选一个子元素。

```
<!ELEMENT choice (good | bad)>
```

上例的choice元素可选一个good子元素，或bad子元素，且只能从选一个。可选的参数列可以多项，不限于两项。如：

```
<!ELEMENT choice (one | two | three | four)>
```

- 小括号，可用小括号把选项括起来，以表达更丰富的意思，如我们想表示choice元素必须包含一个good子元素，并且必须包含ok子元素或bad子元素的一个。

  ```
  <!ELEMENT choice (good，(ok|bad))>
  ```

- 混合内容，在一些文档中，一个元素可能既包含子元素，也包含字符串，这些内容叫混合内容。可用以下方式表示：

  ```
  <!EMEMENT description (#PCDATA | term)* )>
  ```

  该声明表示description元素可包含已析的字符串和term子元素，且允许出现零次或多次，如：

  ```
  <description>
  this is a <term>dtd</term> test.
  </description>
  ```

PCDATA必须在第一位，可选的子元素可任意多项。

- 空元素，某些元素不用包含任何内容，称之为空元素。写成以/>结束的独立标签。

  ```
  <!ELEMENT image EMPTY>
  ```

  示例：

  ```
  <image src="http://www.xml.com/dtd.jpg" />
  ```

- ANY，允许元素内包含任意内容。该选项在dtd测试时很有用，在生产系统中尽量不要使用。

  ```
  <!ELEMENT page ANY>
  ```

3.3. 属性声明

一个有效的XML文档，必须对元素的属性进行声明。使用ATTLIST声明来完成，一个ATTLIST可以为一个元素类型声明多个属性。

```
<!ATTLIST image src CDATA #REQUIRED>
```

上例声明image元素必须有一个src属性，该属性的值是字符数据。可用ATTLIST声明为一个元素声明多个属性，如：

```
<!ATTLIST image src    CDATA #REQUIRED
                width  CDATA #REQUIRED
                height CDATA #REQUIRED
                alt    CDATA #IMPLIED
>
```

上述声明指出src、width、height属性是必须的，alt属性是可选的。

3.3.1. 属性类型

- CDATA类型属性值可包含任意文本字符串。DTD不能指定属性为一个整数或一个日期，Schema能提供更为强大的数据类型。
- NMTOKEN类型属性值是一个XML名称记号。XML名称记号与XML名称类似，但XML名称记号允许所有的字符作为名称的开始字符，而XML名称的第一个字母必须是字母、表意字符和下划线。因此10，.bashrc是合法的XML名称标记，但不是合法的XML名称。每个XML名称都是一个XML名称标记，然而XML名称标记不全是XML名称。如果属性包含1990，2005之类的整数，则应该指定其类型为NMTOKEN。如：

  ```
  <!ELEMENT person birthday NMTOKEN #REQUIRED>
  ```

- NMTOKENS类型属性包含一个或多个用空白分隔的XML名称记号。如：

  ```
  <person dates="02-01-2005 03-01-2005 05-01-2005">person</person>
  ```

  对应的声明应为：

  ```
  <!ATTLIST person dates NMTOKENS #REQUIRED>
  ```

  另一方面，对01/02/2005这样的形式不能使用该声明，因为其中的正斜杠不是合法的名称字符。

- 枚举声明，枚举不用关键字。直接列举所有的值，中间用竖线分隔。如：

  ```
  <!ATTLIST date month(January | February | March | April | May | June | July | August
  ```

  针对上述声明，date元素的month属性可选十二个月份的中一个。

- ID类型的属性必须包含一个XML名称，而且该名称在文档中是独一无二的。ID属性可为元素分配一个唯一的标识符。

```
<!ATTLIST name card_id ID #REQUIRED>
```

由于数字不是合法的XML名称，所以ID编号不能以数字开头，解决办法是在前面加下划线或字母。

- IDREF类型的属性指向文档中某元素的ID类型的属性。因此，它必须是一个XML名称，它的作用是当简单的包含关系不能满足要求时在元素间建立多对多关系。如：

```
<project project_id="p1">
   <goal>deploy linux</goal>
   <team_member person_card_id="c123">
</project>
<person card_id="c123">
   <name>linuxsir</name>
   <assignment project_project_id="p1">
</person>
```

project元素的project_id属性和person元素的card_id属性应该是ID类型。team_member元素的person_card_id属性和assignment元素的project_project_id属性是IDREF类型。对应的声明如下：

```
<!ATTLIST person  card_id    ID #REQUIRED>
<!ATTLIST project project_id ID #REQUIRED>
<!ATTLIST team_member person_card_id     IDREF #REQUIRED>
<!ATTLIST assignment  project_project_id IDREF #REQUIRED>
```

- IDREFS类型的属性包含一个XML名称列表。名称间用空白间隔，且每个名称都是文档中某个元素的ID。当某个元素需要引用多个其他元素时使用该元素。如：

```
<!ATTLIST person card_id    ID     #REQUIRED
                 assignment IDREFS #REQUIRED>
<!ATTLIST project project_id ID     #REQUIRED
                 team        IDREFS #REQUIRED>
```

对应的文档可写成：

```
<project project_id="p1" team="c123">
   <gold>deploy linux</gold>
</project>
<person card_id="c123" assignment="p1">
   <name>Linuxsir</name>
</person>
```

- ENTITY类型的属性包含在DTD的其它位置声明的未析实体的名称中。如movie元素可能有一个标识激活时播放mpeg或rm文件的实体属性：

```
<!ATTLIST movie src ENTITY #REQUIRED>
```

如果DTD声明了一个名为play的未析实体，则此movie元素可用于在XML文档中嵌入视频文件：

```
<movie src="play" />
```

- ENTITIES类型的属性包含在DTD的其它位置声明的多个未析实体名称，其间用空白隔开。

```
<!ATTLIST slide_show slides ENTITIES #REQUIRED>
```

如果DTD声明了未析实体slide1、slide2、slide3、...，则可使用slide_show元素在XML文档中嵌入幻灯片。

```
<slide_show slides="slide1 slide2 slide3" />
```

- NOTATION类型的属性包含在文档的DTD中声明的某个记法的名称。该属性类型较少用。理论上，可以使用该属性使某些特殊元素与类型相关联，下例声明为不同的图像类型定义了4个记法，然后规定每个image元素都必须从中选择一种type属性。

```
<!NOTATION gif SYSTEM "image/gif">
<!NOTATION tiff SYSTEM "image/tiff">
<!NOTATION jpeg SYSTEM "image/jpeg">
<!NOTATION png SYSTEM "image/png">
<!ATTLIST image type NOTATION (gif | tiff | jpeg | png) #REQUIRED>
```

每个image元素的type属性的值可以为gif，tiff，jpeg和png四个值中的一个。该属性比枚举类型稍具优势，因为记法的实际MIME媒体类型在理论上是可用的。由于斜杠在XML名称中不是一个合法字符，所以枚举类型不能指定image/png或image/jpeg作为允许值。

3.3.2. 属性缺省值

每个ATTLIST声明除了要提供一种数据类型外，还要声明属性的缺省行为。

- ## IMPLIED，属性可选。
- ## REQUIRED，属性必须有。
- ## FIXED，属性是常量，不能更改。

```
<!ATTLIST person name CDATA #FIXED "linuxsir"
```

- Literal，作为一个引用字符串的实际缺省值。

```
<!ATTLIST person name NMTOKEN "linuxsir"
```

如果没有显示指明person元素的name属性，则该值为linuxsir。

3.4. 实体

- 用ENTITY声明定义实体。如：

```
<!ENTITY linux "linux is a very good system">
用&linux;可引用该字符串
```

- 可定义一个外部实体，引用外部XML文档

```
<!ENTITY linux SYSTEM "/home/linux/test.xml">
使用&linux;可引用/home/linux/test.xml文档
```

外部实体没有XML声明，但可以有文本声明，两者很类似，主要区别是文本声明必须有编码声明，而版本信息则是可选的。

```
< xml version="1.0" encoding="gb2312" >     是一个合法的文本声明
< xml encoding="gb2312" >                   也是一个合法的文本声明
```

- 不是所有的数据都是XML。如jpeg照片，mpeg电影等。XML建议使用外部未析实体作为在文档中嵌入这些内容的机制。DTD为包含非XML数据的实体指定一个名称和URI。

```
<!ENTITY movie SYSTEM "/home/linux/test.avi" NDATA avi>
```

由于数据不是XML格式，所以使用NDATA声明指定数据类型。avi是在NOTATION中定义的MIME媒体类型。在XML中嵌入未析实体很复杂且不规范，尽量不要使用。

- 参数实体可定义一组通用的实体，在文档中可通过该参数实体来引用实体。参数实体的定义与通用实体定义类似，只是中间多了一个%，引用时也是用%代码&。

```
<!ENTITY % person "name,address,postcode">
引用方法
%person;
这样会用name,address,postcode代替参数实体%person;
```

- 通常DTD都比较大，DocBook的DTD长达11000多行，如果把它存放在单一文件中，管理和维护起来都非常困难。我们可以使用外部DTD子集，把一个大的DTD按功能分成不同的功能块，存放在不同的文件中。再通过外部参数实体声明引入当前DTD中，如：

```
定义参数实体引用外部names.dtd
<!ENTITY % names SYSTEM "names.dtd">
调用外部DTD子集
%names;
```

- 使用IGNORE关键字可注释声明，如:

```
<![IGNORE[
   <!ELEMENT note (#PCDATA)>
]]>
```

当然了，使用<!-- 注释 -->的方式也是一样的。

- INCLUDE关键字表示DTD中的确在使用给定的声明，如：

```
<![INCLUDE[
   <!ELEMENT note (#PCDATA)>
]]>
```

单从该声明来看，有没有使用INCLUDE效果都一样，但如果组合INCLUDE和IGNORE，可实现DTD功能的选择。我们可定义一个参数实体：

```
<!ENTITY % note_allowed "INCLUDE" >
```

然后使用参数实体引用而不使用关键字：

```
<![%note_allowed;[
   <!ELEMENT note (#PCDATA)>
]]>
```

按上述操作，元素声明是有效的，但我们也可以把参数实体%note_allowed重新定义为IGNORE，这样，该元素声明就无效了。

Chapter 4. XML名称空间

- XML名称空间表示XML名称的使用范围，因为XML可自定义元素标签，所以有不同XML应用间XML名称重名的机会是很大的。如果没有一种方法来区分不同应用的名称，就会造成混乱。XML名称空间就是为了解决这个问题而设计的。通过XML名称空间，我们可以区分来自不同的XML应用的具有相同名称的元素和属性。可以将来自单一XML应用的相关元素和属性集合在一起，方便软件识别和处理。

- 名称空间由前缀和本地部分组成，中间用冒号分隔。前缀标识元素或属性的所在名称空间，本地部分标识名称空间中的某个元素或属性。整个名称也称为限定名称(qualified name)。前缀可以用除XML(大小写任意组合)三个字母外的任何合法的XML名称字符组成。每个限定名称中的前缀都必须与唯一的一个URI关联。带有相同URI关联的前缀的名称属于同一名称空间。

```
<rdf:RDF xmlns:rdf="http://www.w3.org/TR/REC-rdf-syntax#">
 <rdf:Description about="http://www.example.com/test.xml">
   <title>example</title>
   <author>linuxsir</author>
   ...
 </rdf:Description>
</rdf:RDF>
```

上例rdf:RDF元素的xmlns:rdf属性将前缀rdf绑定到名称空间 http://www.w3.org/TR/REC-rdf-syntax# 。属性xmlns:rdf为rdf:RDF元素及其子元素声明了前缀rdf。RDF处理器将把rdf:RDF和rdf:Description作为RDF元素，因为两个元素都具有与RDF规范定义的某个URI相绑定的前缀。处理器不会认为title，author等元素为RDF元素，因为它没有绑定到相同URI的rdf前缀。

前缀一般在使用该前缀的最上层元素中定义。在下层元素中也可定义不同的前缀：

```
<rdf:RDF xmlns:rdf="http://www.w3.org/TR/REC-rdf-syntax#">
 <rdf:Description xmlns:dc="http://www.w3.org/dc/"
                  about="http://www.example.com/test.xml">
   <dc:title>example</dc:title>
   <dc:author>linuxsir</dc:author>
   ...
 </rdf:Description>
</rdf:RDF>
```

不带前缀的属性，如about，不属于任何的名称空间。如xlink:type和xlink:href属性属于xlink名称空间，当然，前提是你要先把xlink绑定到一个URI。URI不必须是一定存在的http链接，它只是一种表示的方法，以区分不同的名称空间。

- 通过将无前缀的xmlns属性附加到根元素中，可以指定不带前缀的元素及所有不带前缀的子元素属于某个名称空间。

```
<svg xmlns="http://www.w3.org/2000/svg">
   <ellipse rx="110" ry="130" />
   <rect x="4cm" y="1cm" />
</svg>
```

这里，虽然所有元素都没有前缀，但它都同属一个名称空间。但属性属不同名称空间，因为默认名称空间只应用于元素。默认名称空间在子元素中也用相同的方法重新设置。

- 如果名称空间只用来识别来自某种XML应用的元素和属性，而不是用来区分具有相同名称的不同元素，则可在DTD的元素中定义一个固定的xmlns属性，而不需要文档中定义。定义方法如下：

```
<!ATTLIST svg xmlns CDATA #FIXED "http://www.w3.org/svg/">
```

- 在定义DTD时，需要使用名称空间前缀的在定义时也要把前缀写到DTD定义里，如：

```
<!ELEMENT xlink:name (#PCDATA)>
```

- 使用参数实体引用来定义名称空间前缀可方便DTD文档的维护，如：

```
<!ENTITY % prefix "xlink">
<!ENTITY % colon ":">
```

接着，利用该参数实体名称定义更多的参数实体引用，如：

```
<!ENTITY % xlink-title "%prefix;%colon;title">
<!ENTITY % xlink-author "%prefix;%colon;author">
```

这样，如果需更改前缀，只需修改一个地方就可以了，不用整篇文档修改。

```
<!ELEMENT %xlink-title;  (#PCDATA)>
<!ELEMENT %xlink-author; (#PCDATA)>
```

> 不能在ATTLIST和ELEMENT声明中直接使用%prefix;和%colon；，因为在另一个实体的外部使用这些参数实体时，XML解析器会在实体替换文本的两边添加额外的空格。

Chapter 5. XHTML

XHTML是W3C推荐的一种标准，它定义了一种与XML兼容的HTML版本。XHTML文档是一个有效的XML文档，所以编写格式比HTML严格。如果需从HTML文档转换成XHTML文档，需作以下更改：

- 在XHTML中不允许省略结束标签，所以需补齐缺少的标签。
- 元素需按正确的顺序嵌套。
- 所有元素和属性的名称都采用小写。
- 属性值需添加引号，如<p align="center">。
- 所有属性都需有属性值。
- 采用&和<等的实体形式表示这些字符。
- 确保文档有单一根元素，最好用html。

- 像<hr>这样的空元素要改成<hr/>或<hr><hr/>。
- 注释应由<! 注释 >的形式改成<!-- 注释 -->。
- 文档编码应采用UTF-8或UTF-16，或者添加XML声明指定文档的编码方式。
- 需去掉非标准的元素。如：marguee。
- 添加一个DOCTYPE声明，用PUBLIC来指向XHTML的三种DTD中的一种。分别是Strict、Transitional和Frameset，一般使用Strict。
    - Strict(严格型)，W3C推荐的XHTML形式。不包括一些非标准的元素和属性，如applet和center等。声明方式如下：

    ```
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                         "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
    ```

    - Transitional(过渡型)，一种不太严格的XHTML格式，可使用一些非标准的元素和属性，如applet和bgcolor等。声明方式如下：

    ```
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                         "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    ```

    - Frameset(框架型)，与过渡型DTD类似，允许使用与框架相关的元素，如frameset和iframe。声明方式如下：

    ```
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                         "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
    ```

- 文档的根元素必须具有xmlns属性，标识缺省的名称空间提 http://www.w3.org/1999/xhtml 。

下面是一个标准的XHTML文档的示例：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns:"http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-type" content="text/html; charset=gb2312">
<title>xhtml example</title>
</head>
<body>
...
</body>
</html>
```

由HTML转到XHTML是一种枯燥而乏味的工作，现在有一种叫tidy的开源工具可帮我们完成大部份的工作，它是一个C程序，使用方法如下：

```
% tidy --output-xhtml yes test.html test.xml
```

XHTML 1.1把XHTML的三种DTD分成独立模块。我们可根据实际情况包含或省去某些模块。这些模块是：

- Structure Module(结构模块)--->%xhtml-struct.module;，包含HTML文档主要的元素，如：html、head、title和body。
- Text Module(文本模块)--->%xhtml-text.module;，包含文本的基本元素和其内联元素，如：h1、h2、...、strong、span等。
- Hypertext Module(超链接模块)--->%xhtml-hypertext.module;，包含用于链接的元素，如：a元素。
- List Module(列表模块)--->%xhtml-list.module;，包含用于列表的元素，如：dl、dt、dd、ul、ol和li。
- Applet Module(applet模块)--->%xhtml-applet.module;，Java所需要元素，如：applet和param。
- Presentation Module(表示模块)--->%xhtml-pres.module;，面向表示的标记：b、big、hr、I、small、sub、sup和tt。
- Edit Module(编辑模块)--->%xhtml-edit.module;，用于修正的元素，如：del和ins。
- Bidirectional Text Module(文本方向模块)--->%xhtml-bdo.module;，用于指定文本阅读的方向，如bdo元素。
- Basic Forms Module(基本表单模块)--->%xhtml-basic-form.module;，用于HTML 3.2的表单元素，如：form、input、select、option和textarea。
- Forms Module(表单模块)--->%xhtml-form.module;，用于HTML 4.0的表单元素，如：form、input、select、option、textarea、button、fieldset、label、legend和optgroup。
- Basic Tables Module(基本表格模块)--->%xhtml-basic-table.module;，基本的表格元素，如：table、caption、th、tr和td。
- Table Module(表格模块)--->%xhtml-table.module;，安全功能的表格支持，如：table、caption、th、tr、td、col、colgroup、tbody、thead和tfoot。
- Image Module(图像模块)--->%xhtml-image.module;，包含img元素。
- Client-side Image Map Module(客户端图像映像模块)--->%xhtml-csismap.module;，包含map和area元素以及支持客户端图像映像所需要的元素的属性。
- Server-side-Image Map Module(服务器端图像映像模块)--->%xhtml-ssismap.module;，该模块没有添加新元素，但对img元素添加了一个ismap属性。

- Object Module(对象模块)--->%xhtml-object.module;，用于在网页中嵌入可执行内容，如：java程序。
- Param Module(参数模块)--->%xhtml-param.module;，网页中可执行内容中传递参数的param元素。
- Frames Module(框架模块)--->%xhtml-frames.module;，包含实现框架所需的元素，如：frame、frameset和noframes。
- Iframe Module(内联框架模块)--->%xhtml-iframe.module;，包含内联框架的iframe元素。
- Intrinsic Events(固有事件模块)--->%xhtml-events.module;，支持如onSubmit和onFocus等脚本的属性。
- Meta-information Module(元信息模块)--->%xhtml-meta.module;，包含meta元素。
- Scripting Module(脚本模块)--->%xhtml-script.module;，支持JavaScript等脚本。
- Stylesheet Module(样式表模块)--->%xhtml-style.module;，用于定义CSS的style元素。
- Link Module(链接模块)--->%xhtml-link.module;，指定外部文件，如样式表、库等关系的link元素。
- Base Modue(基模块)--->%xhtml-base.module;，包含base元素，指定解析相对URL所参照的基URL。
- Target Module(目标模块)--->%xhtml-target.module;，用于指定目标框架或框架中某个窗口的target属性。
- Style Attribute Module(样式属性模块)--->%xhtml-inlstyle.module;，将CSS样式应用于文档中单个元素的style属性。
- Name Identification Module(名称标识模块)--->%xhtml-nameident.module;，name属性是id属性的早期版本，现在不推荐使用。
- Legacy Module(传统模块)--->%xhtml-legacy.module;，不推荐使用的元素和属性，如：basefont、center、fonts、strike和u元素。
- Ruby Module(Ruby模块)--->%xhtml-ruby.module;，东亚文本中用于将少量文本放于正文文本旁边的ruby、rbc、rtc、rb、rt和rp元素，一般用来指示发音。

Chapter 6. 样式表

样式表可帮我们解释XML文档中各元素的具体意思，所以通过样式表可直接在浏览器上显示XML文档。目前主要的样式表语言有：

- CSS1(Cascading Stylesheets Level 1，层叠式样式表1)

- CSS2(Cascading Stylesheets Level 2，层叠式样式表2)
- XSLT(XSL Transformations 1.0 XSL 转换 1.0)

在XML文档在序言部分通过xml-stylesheet处理指令可指定关联的样式表。xml-stylesheet指令必须有一个href属性和type属性。href指向样式表的URL，type指定样式表的MIME类型：对CSS为text/css，对于XSLT为text/xml或application/xml。下面是一个简单的使用样式表的XML文档：

```
< xml version="1.0" >
< xml-stylesheet href="test.css" type="text/css" >
...
```

除以上两个必须的属性外，还有4种可选属性：

- media，标识该样式应用于什么媒体，如报纸(paper)、计算机监视器(screen)、电视(tv)或所有(all)。
- charset，指明样式表采用字符集编码方式，如：utf-8。
- alternate，指明是否有可选的样式表，默认为no，表明是主样式表，如果为yes，则是备用样式表。
- title，在有alternate的前提下，title用于指定不同样式表的标题。如：

```
< xml-stylesheet href="big.css" type="text/css" alternate="yes" title="Large fonts" >
< xml-stylesheet href="small.css" type="text/css" alternate="yes" title="Small fonts'
< xml-stylesheet href="medium.css type="text/css" title="Normal fonts" >       #默认的
```

样式表现在已成为Web应用中的一个关键技术，它的作用主要体现在以下三个方面：

- 设计一个样式表可以应用于多个文档。样式表可以存在于XML文档外，XML文档可通过链接使用样式表。这意味着如果你有几千个文档，都可以链接到同一个样式表中，改变一个样式表等于改变几千个文档的显示效果。
- 实现内容和表现的分离，增强文档的一致性和可维护性。通过单一的样式表，实现所有文档显示的一致。如果显示样式有变动，我们只需维护有限的几个样式表就可以了。
- 实现一个文档，多个样式。通过样式表，可把一篇文档以HTML形式、PDF形式或文本形式显示出来。

6.1. CSS2

CSS2是层叠样式表，它是一种排版技术，能让元素按特定的样式显示，如字体大小，颜色、布局等。在网页中有三种使用方法：

- 用<style>标记声明，如

```
<style>
div {font-size: 12pt;}
div {color: blue;}
</style>
```

- 在元素中用style属性指定，如：

```
<div style="font-size: 12pt;color: blue">CSS测试</div>
```

- 用LINK标记链接一个外部CSS文件，如：

```
<link rel="stylesheet" type="text/css" href="mycss.css">
```

按作用域来分，有三类的样式表，分别是网页解释器样式表、作者样式表和浏览者样式表。网页解释器样式表也叫默认的样式表，当没有另外的样式表加载时使用。作者样式表就是网页设计师设计的样式表。浏览者样式表是浏览网页的用户在浏览器上另外设置的样式表。

CSS的基本数据类型

- integer，表示整数，可取正负值。如：12，-24。
- number，表示数字，可取正负值和小数。如：12.1，-14.3。
- lenght，表示距离长度，可取正负值和小数，后跟一个单位，如:12em，12cm。单位又分相对单位和绝对单位，相对单位有：em，ex，px。绝对单位有：in(英寸)，cm(公分)，mm(公厘)，pt(等于1/72英寸)，pc(等于12pt)。
- percentage，表示百分比值，可取正负和小数。如：20%，-40%。
- uri，表示网络资源。如：http://www.ringkee.com 。

inherit参数值

```
<style>
body {width: 600px;}
.div1 {width: 120%;}
.div2 {width: inherit;}
说明：
div1的宽度是600px*120%
div2的宽度继承父元素body的参数，是600px
```

选择符的作用是指定哪些元素使用哪些样式。选择符可以分为简单选择符和复合选择符两类，简单选择符是类型选择符、通用选择符加上零个或多个属性选择符、ID选择符、伪类等组成。复合选择符是用">"和"+"号结合多个简单选择符组成。">"和"+"号两边要加上空格。下面介绍各种选择符：

- 通用选择符，用"*"号表示，可用于所有标记。如：

```
<style>
* {font-size: 14pt;}
*.EM {color: red;}
</style>
<div>应用字体样式</div>
<em class="EM">应用红色样式</em>
```

- 类型选择符，与标记名一样，只作用已该标记上。如：

```
<style>
div {font-size: 14pt;}
</style>
<div>应用样式</div>
```

- 子代选择符，HTML标记是可嵌套的，子代选择符可把样式表应用于子嵌套的子标记上，如：

```
<style>
div p b {font-size: 14pt;}
</style>
<div>
<p>没有应用样式</p>
<p><b>应用样式</b></p>
</div>
```

- 子选择符，与子代选择符类似，但它只调用第一层子元素。如：

```
<style>
div > b {color: red;}
div p > em {color: green;}
</style>
<div><b>当b标记是div标记的子标记时应用红色样式</b></div>
<div><p><em>当em是p的子标记且p是div的子标记时应用绿色样式</em></p></div>
```

- 邻近选择符，当两个元素位于同一层且在位置是前后关系时，可以使用邻近选择符。两个选择符用"+"号分开，如果A位于B之前，则B可应用样式。如：

```
<style>
div + p {color: red;}
</style>
<div>没有应用样式</div>
<p>应用红色样式。</p>
```

- 属性选择符，HTML标记有属性，我们可为特定的属性指定样式。有四种写法，分别是：
    - [属性]，样式只应用于指定的属性。
    - [属性=值]，样式只应用于指定的属性与值都相同的情况

- [属性~=值]，样式只应用于指定的属性且属性值包含指定值的情况，属性值是用空格分隔的字符串。
- [属性|=值]，样式只应用于指定的属性且属性值是的第一个字符串是指定值的情况，属性值是用"-"分隔的字符串。

```
<style>
[href] {color: red;}
A[href="http://www.ringkee.com"] {color: green;}
table[summary~="table"] {color: black;}
table[summary|="this-is-a-table"] {color: blue;}
</style>
<a href="http://www.python.org">应用红色样式</a>
<a href="http://www.ringkee.com">应用绿色样式</a>
<table summary~="This is a table>
 <tr>
  <td>应用黑色样式</td>
 </tr>
</table>
<table summary|="This-is-a-table>
 <tr>
  <td>应用蓝色样式</td>
 </tr>
</table>
```

- 类选择符，与属性选择符类似，但它只指对class属性应用样式。类选择符用"."语法，如.value与[class~=value]是一样的。

```
<style>
.myid {color: red;}
</style>
<div class="myid">应用红色样式</div>
```

- ID选择符，与属性选择符类似，但它只指对ID属性，用"#"语法。

```
<style>
#myid {color: red;}
</style>
<div id="myid">应用红色样式</div>
```

- :first-child伪类，当标记是另一个标记的第一个子标记时，应用样式。

```
<style>
p:first-child {color: red;}
</style>
<p>p是body的第一个子标记，应用红色样式</p>
<div>测试</div>
<p>p标记是body的第三个子标记，不应用红色样式</p>
```

- :link和:visited伪类只作用于a标记，在指定href属性的前提下，:link表示a标记还没被点击时的样式，:visited表示被当点后的样式。

```
<style>
a:link {color: blue;}
a:visited {color: red;}
</style>
<a href="http://www.ringkee.com">链接没点击前是蓝色的，点击后是红色的</a>
```

- :hover，:active和:fouce伪类也只能作用于a标记，且也要指定href属性。:hover指定当用户把鼠标移到a标记上并且指针变成手型时应用的样式。:active指定点击a链接并放开鼠标时所显示的样式。:fouce指定用户点击a标记瞬间，即链接成为焦点时所显示的样式。:hover要放在:link和:visited之后，否则:hover的样式会覆盖:link和:visited的样式。

```
<style>
a:link {color: blue;}
a:visited {color: red;}
a:haover {color: green;}
a:focus {color: black;}
a:active {color: white;}
</style>
 <a href="http://www.ringkee.com">应用样式</a>
```

- :left及:right伪类只作用于页面内容。当页面在左边时应用:left指定的样式，当页面在右边时应用:right指定的样式。
- :first-line只对div和p标记不效，样式只应用于这两个标记内的第一行内容。

```
<style>
:first-line {color: red;}
</style>
<div width:50px;>
该元素内的第一行内容应用红色样式。
</div>
```

- :first-letter伪类也只能作用于div和p标记，与:first-line不同的是它只作用于标记内的第一个字符。如果我们想要每一行的开头字符大一点就可使用该伪类。

```
<style>
:first-letter {font-size: 40pt;}
</style>
<p>这行文字开头第一个字符的大小是40pt</p>
```

- :before和:after伪类可在内容的前面或后面增加特定的内容或指定样式。

```
<style>
p:before {content: "("; color: red;}
p:after {content: ")"; color: green;}
</style>
<p>这行文字前后会增加一对括号，前括号为红色</p>
<p>这行文字前后会增加一对括号，后括号为绿色</p>
```

- 层叠选择符是指当有多个选择符的样式都应用于同一个标记时的选择规则。该规则利用一个三位数来确定，数字最大的就可选中。这三位数的确定规则的这样的，如果选择符中有ID选择符，则百位数加1,否则为0。如果有属性选择符、类选择符或伪类选择符，则十位数加1，否则为0。如果有类型选择符，则个位数加1，否则为0。如果选择符是#div div，这三位数则是101。让我们分析一下，#div是ID选择符，所以在百位数上加1，div是类型选择符，所以个位数上加1变成101。"*"表示0，优先级最低。

样式表的主要功能是指定同一个文件在不同媒体上按不同的样式显示。通过在种方式可指定不同媒体

- @media方式

  ```
  <style>
  @media screen {div{color:red;}}
  @media print {div{color:green;}}
  </style>
  <div>不同媒体显示不同颜色</div>
  ```

- @import是另一种指定不同媒体的方式，它可引入外部的css文档。它的语法格式是：

  ```
  <style>
  @import url("simple.css") screen;
  </style>
  ```

- 在HTML4.0中，可以用LINK标记的media属性为不同媒体类型指定样式表。

  ```
  <LINK rel="stylesheet" href="import.css" type="text/css" media="print">
  ```

!important规则会改变应用样式的优先级，有!important参数样式的优先级最高，会优先显示。

```
<style>
h1 {color:red;}
h1 {color:green !important;}
</style>

<h1>字体为绿色</h1>
```

6.2. XSLT

XSLT是XSL的一部份，它是XML的一种应用，指定将一篇XML文档转换成另一种XML文档的规则。XSLT文档即是一篇XML文档，也是一个样式表，里面包含一系列的模板。XSLT处理器对输入XML文档中的元素和样式表中的模板进行比较，如果匹配，则将该模板的内容写入一个输出树中。完成处理后，将输出树串行化成一篇XML文档或其它格式的文档，如HTML或者rtf。

XSLT几个关键术语

- 源树，原始文档中的元素和元素内容的树。
- 结果树，转换之后中文档中的元素和元素内容的树。
- 模板规则，XSLT样式表的基础，分为模式和模板两部份。整个xsl:template元素。
- 模式，表示源树中的元素与模式规则匹配的条件集合。xsl:template中的match的值。
- 模板，表示当应用模板规则时，结果树中要实例化的部份。xsl:template元素中的内容。

XSLT定义了35个元素，分为三类：

两个根元素

- xsl:stylesheet根元素，XSLT也是一个XML文档，该文档的根元素就是xsl:stylesheet。XSLT元素都属于名称空间 `xmlns:xsl="http://www.w3.org/1999/XSL/Transform"` ，所以所有的XSLT元素都有xsl前缀。一个最小化XSLT文档：

  ```
  < xml version="1.0" >
  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  </xsl:stylesheet>
  ```

- xsl:transform元素，作用同上。

13个顶级元素，可直接作为根元素的子元素，包括：

- xsl:apply-imports
- xsl:attribute-set
- xsl:decimal-format
- xsl:import
- xsl:include
- xsl:key
- xsl:namespace-alias
- xsl:output
- xsl:param
- xsl:preserve-space
- xsl:strip-space
- xsl:template模板元素，用于匹配XML文档中的元素。如：<xsl:template match="person">，匹配XML文档中的person元素。

- xsl:variable

20个指令元素

- xsl:apply-imports
- xsl:apply-template应用模板元素，用于显示指定的元素值(内容)。如：<xsl:apply-template select="name">，显示name元素的值。
- xsl:attribute
- xsl:call-template
- xsl:choose
- xsl:comment
- xsl:copy
- xsl:copy-of
- xsl:element
- xsl:fallback
- xsl:for-each
- xsl:if
- xsl:message
- xsl:number
- xsl:otherwise
- xsl:processing-instruction
- xsl:text
- xsl:value-of选择元素，用于计算元素的值(内容)。如：<xsl:value-of select="name">，获得XML文档中name元素的值(内容)。
- xsl:variable
- xsl:when

XSLT函数

6.3. XPath

XPath是一种用来从文档树中选择节点和节点集的语言。从XPath的角度来看，共有七种节点：

- 根节点
- 元素节点
- 属性节点
- 文本节点
- 注释节点
- 处理指令节点
- 名称空间节点

CDATA部份，实体引用和文档类型声明不包括在内，XPath在所有这些项都并入文档之后才起作用。根节点和根元素是不同的两个概念，根节点包含整篇文档，包括根元素。

6.3.1. 匹配模式

匹配模板的通用模式

Table 6.1.

模式	描述
match="E"	匹配元素E
match="*"	匹配任意元素
match="E|F"	匹配元素E和F
match="E/F"	匹配以E为父元素的元素F
match="E//F"	匹配以E为根元素的元素F
match="/"	匹配根节点
match="text()"	匹配文本节点
match="comment()"	匹配注释节点
match="processing-instruction()"	匹配处理指令
match="node()"	匹配除属性节点和根节点外的节点
match="id(test)"	匹配具有唯一ID test的元素
match="E[@CLASS="foo"]	匹配元素E，其类属性为foo
match="E[F]"	匹配元素包含有F元素的E元素

6.3.2. XPath轴

XPath提供了选择节点的机制，两个较有用的是轴选择和谓语选择，轴指定上下文节点和要选择的节点的关系。共有十三种轴，最常用的有四种，分别是子轴(child)、属性轴(attribute)、自己(self)、双亲(parent)。

Table 6.2. XPath轴描述

轴	描述
child	包含当前节点的儿子
descendent	包含当前节点的后代，后代不包含属性(attribute)或名称域(namespace)节点
parent	包含当前节点的父亲
ancestor	包含当前节点的祖先，祖先总是包含根节点
following-sibling	包含当前节点随后的所有节点树，但不包含attribute或namespace节点
preceding-sibling	包含当前节点之前的所有节点树，但不包含attribute或namespace节点
following	包含当前节点随后的所有节点，following轴排除了当前节点的后代和attribute或namespace节点
preceding	包含当前节点之前的所有节点，following轴排除了当前节点的后代和attribute或namespace节点
attribute	包含当前节点的所有属性
namespace	包含当前节点的所有namespace节点
self	只包含当前节点
descendent-or-self	包含当前节点和当前节点的后代
ancestor-or-self	包含当前节点和当前节点的祖先

6.3.3. 谓词

XPath表达式可以匹配多个节点，如需对匹配的节点进行进一步的筛选，可以使用谓词。

Table 6.3. 选择节点常用谓词

谓词	描述
select="E"	选择是当前节点的孩子的E元素
select=""	选择当前节点的孩子的所有元素
select="text()"	选择当前节点的文本节点孩子
select="@name"	选择当前节点的name属性
select="@*"	选择当前节点的所有属性
select="E[1]"	选择当前节点的孩子的第一个E元素
select="E[last()]"	选择当前节点的孩子的最后一个E元素
select="*/E"	选择当前节点的孙了的所有E元素
select="E//F"	选择从当前节点的孩子的E元素派生而来的元素F
select="//"	选择根元素
select="//E"	选择从根节点派生而来的E元素
select="//E/F"	选择所有是从根节点派生而来的E元素的孩子的F元素
select="."	选择当前节点
select=".//E"	选择从当前节点派生而来的所有E元素
select=".."	选择当前节点的父节点
select="../@name"	选择当前节点的父节点的name属性
select="E[@name='foo']"	选择所有是当前节点的孩子，并且其name属性具有foo值的E属性，除等号外，还可用<，>，<=，>=和!=
select="E[@foo and @bar]"	选择所有包含foo和bar属性的E元素

home/person/@id这种定位路径的写法叫简写定位路径，该写法简洁，容易理解，是XSLT匹配模式中最常用的写法。还有一种称为非简写定位路径的写法，它把节点测试和轴结合在一起，如child::home/child::person/attribute::id。该写法在实际使用中不常用，但它具有非常重要的性能因此有必要了解。

6.3.4. XPath表达式

位置路径是XPath的一个最常用的表达式，用以标识XML文档的节点集。除此之上，XPath表达式还可返回数字、布尔和字符串。非节点集的XPath表达式不能用于xsl:template元素的match属性中。它们用于xsl:value-of元素的select属性值或用于位置路径的谓词中。

每个XPath位置路径可分为一步名多步，每步以“/”号分隔，如：

```
room[\@name=$root]/date[year=$year and month=$month]/meeting
```

上下文节点即当前正在处理的节点，也就是位置路径定位的当前节点。上下文在XPath表达式计算前被创建，由XSLT处理器创建。处理每一步后，上下文都会改变。

位置路径中的步可分为三部份：轴(axis)、节点测试(note test)和谓词(predicate)，它的写法如下：

```
axis::note-test[predicate]
```

轴和节点测试之间用“::”分开，每个谓词由括号〔〕括起来。

要设计好一个位置路径，需确保在每一步选择最少的节点，使用最严格的轴，用最严格的节点测试。避免使用谓词，因为由轴和节点测试选择的节点集的每个节点都会用作谓词的上下文节点。对于位置路径的三步，最节省的是节点测试。

XPath中的所有数字都是8个字节的IEEE754浮点双精度类型，与java的double类型相同。可表示正无穷大、负无穷大和NaN(零除零)值。支持五种运算符，分别是加(+)、减(-)、乘(*)、除(div)、取余(mod)。

XPath中的字符串是Unicode字符，用单引号或双引号定界。可以使用=和!=对字符进行比较，也可用<，>，<=，>关系运算符，但比较的两个字符必须是数字，否则比较结果没有意义。

XPath中的布尔值常用于位置路径的谓词中，如/person[name="debian"]。布尔值还常用于xsl:if和xsl:when元素的test属性中。如：

```
<xsl:template match="home">
   <xsl:if test = ".='debian' or .='redhat'">
      <xsl:value-of select = "." />
   </xsl:if>
</xsl:template>
```

6.3.5. XPath函数

XPath还提供很多函数，用于表达式和谓词。XPath函数的返回值有四种类型，分别是：

- 布尔值，如：true()返回ture(真)，false()返回false(假)，not()对布尔值取反。
- 数字，如：number()把任意类型转化数字，celing()返回大于或等于参数的最小整数。
- 节点集，如：position()返回当有节点在上下节点列表中的位置，count()可统计节点数。
- 字符串，如：string()转化任意类型为字符串，string-length()返回字符串长度。

6.4. XLink

XLink是一种基于属性的语法，用来在XML文档中添加链接。XLink链接可以是单向的，如HTML中的A元素，它也可以是双向的，在两个方向上链接两篇文档，因此能够从A到B或从B到A。每个XLink元素必须具有一个xlink:type属性，指出连接类型。属性xlink:href指向所链接的资源URI。下面是一个简单链接的示例：

```
<test xmlns:xlink = "http://www.w3.org/1999/xlink"
      xlink:type = "simple"
      xlink:href = "http://www.ringkee.com/xml.html">
<author>Jims</author>
<date>2005/02/18</date>
</test>
```

xlink:type属性类型共有六种，分别是：simple，extended，locator，arc，title，resource。

xlink:show属性可告诉浏览器或应用程序在激活链接时应该做什么，它有五种可能的动作，分别是：

- new，在新窗口中显示链接内容。
- replace，在当前窗口显示链接内容。
- embed，在当前链接元素的位置嵌入内容。
- other，动作不确定，由应用程序指定。
- none，无动作。

xlink:actuate属性可告诉浏览器何时显示链接，它有四种可能值：

- onLoad，一旦发现链接，马上显示。
- onRequest，当用户提出请求时才显示。
- other，由文档中的其它标记，而不是xlink，来决定何时显示。
- none，不指定。

一个和HTML中的A元素作用一样的示例：

```
<test xmlns:xlink = "http://www.w3.org/1999/xlink"
      xlink:type = "simple"
      xlink:href = "http://www.ringkee.com/xml.html"
      xlink:actuate = "onRequest"
      xlink:show = "replace" >
<author>Jims</author>
<date>2005/02/18</date>
</test>
```

一个在页面嵌入图像的示例：

```
<image xlink:type = "simple"
       xlink:actuate = "onLoad"
       xlink:show = "embed"
       xlink:href="http://www.ringkee.com/flower.png"
 width = "320" height = "240" />
```

xlink:actuate和xlink:show是可选的。

xlink:title和xlink:role属性可指定资源之间的描述，xlink:title包含少量描述远程资源的文本，xlink:role包含URI，指向资源的较长描述。

Chapter 7. 分析XML

分析XML文档可通过程序来做，分析器有两大类，一种是事件驱动的，一种是基于树模型的。

- 使用事件驱动的分析器时，每遇到一个元素就会触发一个事件，由事件处理器进行处理。事件分析器按顺序读取XML文档，而不把整个文档读入内存，所以处理速度很快。但缺点是由于要从头到尾读取XML文档，因此无法在XML文档中移动位置。事件驱动分析器适合处理其它地方使用的XML数据，如转换成HTML文档或从文件中读取数据并插入数据库中。它的优点有：
  - 文件搜索，从XML文档中搜索需要的标志或数据；
  - 格式转换，如转换成HTML。任何需将原始XML转换成另一种格式的工作都最好使用事件驱动分析器来完成，因为它可动态将信息转换成新格式。
  - 少量修改，你可用事件驱动分析器读取和重新生成XML。在分析过程中，可以改变少量的单语、字符数据内容或重新构造XML。事件驱动分析器特别适合整理和重新格式化XML文档。
  - 简单验证，由于整个文档不在内存中，所以无法进行完整验证，但可检查拼写错误和一般良构XML文档之类的简单问题；
  - 建立内部结构，可以使用事件驱动分析器建立XML文档的复杂内部表示，如基于树的接口使用的树式结构。

  > 事件驱动分析器不能在XML文档间交叉引用文档内容，但它使用简单，速度快。

- 基于树的分析器把整个XML文档读入内存，并生成树状结构。分析器可随机访问树中的任意节点，并能修改树结构和内容。

7.1. 分析器工具

现有的分析器种类有上百种，但常用的是两个标准的工具库，一个是XML简单API(SAX，Simple API for XML)和文档对象模型(DOC，Document Object Model)。SAX是事件驱动分析器的标准，而DOM是基于树的分析器标准。另外，Expat虽然不是标准，但它是脚本语言中处理XML时最常用的分析器。Expat由James Clark编写，是事件驱动分析器。

7.2. Unicode

计算机并不能正真理解文本内容，它无法识别诸如a,b,c这类的字母，更不用说中文了。计算机所能理解的只有数字，如60，80等。字符集(character set)规定了字母到数字的映射关系，如65代表大写字母A。65称为码点(code point)，字符编码(character encoding)决定码点如何用字节表示。是用多了节还是单字节，高字节位表示什么，低字节位表示什么。

不同国家使用不同的语言，不同程序使用不同的编码规范，在进行世界范围内的数据交换就要统一表示数据的字符编码规范。传统的ASCII字符集只定义了127个字符，其中前31个是控制符。127位之后的字符随平台不同而不同。大多数平台只能表示前127位，单字节(8位)，使得字符集中最多只能提供256个字符。这些标准字符称为罗马或拉丁字符集，用ASCII来表示中文、日文是远远不够的。

为了解决字符集问题，出现了Unicode字符集。它可用多字节格式编码字符，目前标准允许2字节字符，支持65536个不同字符。标准的Unicode字符集为Latin-1(或ISO-8859-1)。有关Unicode的介绍可访问Unicode的官方网站：http://www.unicode.org。

Unicode字符集为字符分配码点，即编号。这些编号可以用多种模式编码，如UCS-2、UCS-4、UTF-8、UTF-16。

- UCS-2，也叫ISO-10646-UCS-2。每个字符用一个0~65535之间的两个字节的无符号整数表示。如A的Unicode码点为65，用两个字节00和41(十六进制)表示。B的Unicode码点为66，用两个字节00和42表示。UCS-2有两种形式：高字节(#x0041)在前和低字节(#x4100)在前。为区发高低位不同表示形式，采用UCS-2编码文档通常以Unicode字符#xFEFF(零宽度无间断空格)开头，一般称为字节顺序标记(byte order mark)。这个字符是不可见的。如果两个字节交换位置，得到的字符#xFFFE实际是不存在的。因此中通过查看UCS-2文档的前两个字符是#xFEFF还是#xFFFE，就可确定该文档是否是高字节在前。UCS-2的缺点：如果文本字符主要是拉丁文，由于采用两个字节，字符集编码是单字节字符编码的两倍；UCS-2不能与ASCII向前或向后兼容，用于单字节字符集的工具常常不适用于处理UCS-2编码文件。

- UTF-8是一种可这长度的Unicode编码。0~127为ASCII码字符集，与ASCII编码完全兼容，每个字符采用一个字节编码。UTF-8用两个字节表示128~2047，该范围覆盖了最常见的非表意字母。其余的字符，主要来自汉语、日语和韩语，每个都用3个字节表示。如果Unicode的码点超过65535个字符，那么这些字符就会用4个字节编码。对于以拉丁文

为主的文件，使用UTF-8比UCS-2可减少一半的文件大小。对于汉语、日语和韩语的文件，其大小会增加百分之五十。对于其它语言，文件大小相差不大。UTF-8是最常用的Unicode编码方式。

在Unicode流行以前，出现了一系列处理特定语言的单字节字符集，ISO将14种这样的字符集标准化成ISO 8859标准，分别是ISO-8859-1~14。ISO-8859-15是ISO-8859-1的修订版本。这些字符集统称ISO字符集。

Cp1252是依赖于Windows平台的一种编码，是Windows的缺省字符集。该种编码不支持跨平台特性，尽量不要使用。

MacRoman是Mac OS使用的一种非标准、单字节编码。在非Mac平台下使用也会有问题，尽量不要使用。

在XML文档中，如果需输入编辑器不支持的字符，我们可用字符引用的方式，以十进制或十六进制给出它所代表的Unicode字符编号，如њ(十进制)或者&#x45A(十六进制)。字符引用可用于元素内容、属性和注释，不能用于元素名和属性名、处理指令或XML关键字。如果有一些字符需经常使用，则我们可为这些字符定义实体，这样，在文档中就可方便地引用该实体了。专门定义字符实体的DTD我们可独立出来，形成以.ent为后缀的外部DTD。在需要时使用外部参数实体引用将这些定义引入文档的DTD中。

XHTML 1.0 DTD包含有三个有用的字符引用实体可在文档中使用。

- Latin-1字符，http://www.w3.org/TR/xhtml1/DTD/xhtml-lat1.ent

  ISO-8859-1中自160以上的非ASCII码字符。

- 特殊字符，http://www.w3.org/TR/xhtml/DTD/xhtml-special.ent

  ISO-8859-2中不在Latin-1中的字母。

- 标点符号，http://www.w3.org/TR/xhtml-symbol.ent

  希腊字母表(不包含带重音的字符)和各种标点符号、数学运算符及其他数学中常用的符号。

在XML文档中可以使用xml:lang属性规定元素内容采用的语言。这样就可在一篇文档中同时使用多种语言，这是XML跨平台和跨语言的重要特性之一。如：xml:lang="CN-CHN"。语言代码是一个两个字母的语言代码，语言代码后还可跟一个子代码，语言代码可在这里找到http://ftp.ics.uci.edu/pub/ietf/http/related/iso3166.txt。下面是xml:lang属性声明的示例：

```
<!ELEMENT test (#PCDATA)>
<!ATTLIST test xml:lang NMTOKEN #IMPLIED>
```

由于所有语言代码都是有效的XML名称标记，所以使用NMTOKEN类型。

Appendix A. 附录

A.1. 标记语言的历史

- 1974年，IBM的Charles F.Goldfarb、Ed Mosher和Ray Lorie发明了一种最终发展成为标准的通用标记语言(SGML，Standardized General Markup Language)。1986年，SGML被ISO采用为8879号标准。
- 1991年，Tim Berners-Lee利用SGML提供的基本机制定义了超文本标记语言(HTML，Hypertext Markup Language)，把Internet带进了一个多姿多彩的世界。HTML是SGML最成功的一种应用。
- 1998年2月10日，W3C的XML 1.0标准正式发布，为异构平台的数据交换提供了一个可行的标准。

A.2. XML相关技术名词解释

- XLink，一种基于属性的用于XML和非XML文档之间超链接的语法，可以提供与HTML相似的简单、单向的链接，多文档之间的多向链接，以及没有写入权限的文档间的链接。
- XSLT(XSL Transformation,XSL转换)，一种XML文档，用于描述具有相同或不同XML词汇表的两个文档之间的转换。
- XSL-FO(XSL Formatting Object,XSL格式化对象)，一种用于描述打印和网页布局的XML应用。
- Dsssl(Document Style Sheet and Semantics Language,文档样式表和语义语言)，用于描述XML打印和在Web上的样式，源自SGML。
- XPointer，标识XML文档中由URI指定的特殊部分，通常与XLink结合使用。
- XPath，用于标识XML文档中的路径。
- Namespace，区分来自不同的XML语汇表但名称相同的元素和属性的一种方式。
- SAX，Simple API for XML，一种事件驱动的XML文档处理器。
- DOM，Document Object Model，一种面向树的API，它把XML文档解释成具有多属性和嵌套的对象树。

A.3. XML应用

XML可自定义标签，为了增强互操作，个人或组织可约定只使用某种标签。这些标签集被称为XML应用。XML应用不是使用XML的软件应用程序，如IE或Word，而是XML在矢量图形或数学公式这些特殊领域的一种应用。

- SVG，可缩放矢量图形(Scalable Vector Graphic)，是W3C推荐的XML中矢量图的编码标准。
- MathML，数学标记语言(Mathematical Markup Language)，用于表示数据公式。
- CML，化学标记语言(Chemical Markup Language)，描述化学、物理学、分子生物学和其它分子科学。
- RDF，资源描述框架(Resource Description Framework)，用于描述资源，特别是图书馆分类卡上的元数据。
- CDF，通道定义格式(Channel Definition Format)，微软公司定义的一种非标准XML应用，用来向IE发布可离线浏览的网页。